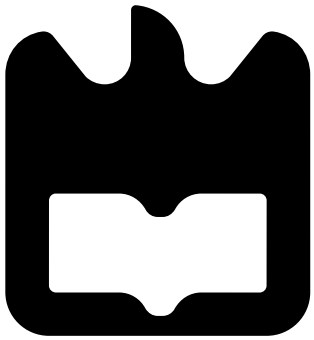
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2012

Bruno Miguel
Sendas Parreira

INTEGRAÇÃO DA CLOUD COM REDE NA PERSPECTIVA DE OPERADOR



Bruno Miguel Sendas Parreira

INTEGRAÇÃO DA CLOUD COM REDE NA PERSPECTIVA DE OPERADOR

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica da Professora Dra. Susana Sargento , Professora Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Mestre Jorge Carapinha, Engenheiro da Portugal Telecom Inovação.

o júri / the jury

presidente / president

Professor Doutor Anibal Manuel de Oliveira Duarte
Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Rui Manuel Rodrigues Rocha
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores do Instituto Superior Técnico da Universidade Técnica de Lisboa

Professora Doutora Susana Isabel Barreto de Miranda Sargento
Professora Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Mestre Jorge Manuel dos Santos Correia Carapinha
Engenheiro do Departamento de Coordenação Tecnológica e Investigação Exploratória da Portugal Telecom Inovação

Agradecimentos / Acknowledgments

Esta dissertação veio concluir uma fase importante do meu percurso académico.

Tal não seria possível sem o apoio incondicional da minha mãe e da minha família que estiveram sempre presentes nos momentos mais importantes. Obrigado por tudo.

Quero agradecer também à Prof. Dra. Susana Sargento e ao Engenheiro Jorge Carapinha pelo apoio, conselhos e pelo modo como souberam liderar o grupo de trabalho no qual estive inserido. Aos Mestres Márcio Melo e João Soares pela ajuda, partilha de conhecimento, companheirismo e pela maneira como fui integrado neste projecto. Quero também dar uma palavra de apreço a todas as pessoas que se cruzaram comigo no Instituto de Telecomunicações, pela alegria e disponibilidade que trouxeram para este local de trabalho.

À minha melhor amiga e namorada, Sara, e sua família pelo apoio, compreensão e acolhimento.

A todos os meus amigos, que sempre confiaram nas minhas capacidades para superar este desafio, pelos momentos de descompressão e motivação.

Resumo

Cloud Computing (CC) tem sido nos últimos tempos um tema bastante mediático no mundo da tecnologia, sendo claras as suas potencialidades tanto em termos económicos como em eficiência de recursos. Apesar destas valências, a adopção massiva deste paradigma está ainda condicionada por aspectos relacionados com interoperabilidade, segurança e Quality of Service (QoS). É no sentido de colmatar estas condicionantes que surge o conceito de Cloud Networking, que consiste na integração das características de CC na rede.

Actualmente a infra-estrutura de rede é baseada em tecnologias que foram desenhadas há muitos anos, numa altura em que o contexto e as necessidades do mundo das comunicações eram completamente distintas das necessidades actuais. Este fenómeno tem sido bastante abordado e referido como a ossificação da rede. Esta infra-estrutura é, actualmente, incapaz de dar resposta às necessidades do CC, características como a dinâmica e flexibilidade não se reflectem na rede. No entanto, a rede e CC não podem ser dissociados, pois é esta que estabelece uma ponte e permite o acesso a estes recursos por parte dos utilizadores influenciando negativamente o seu uso.

No futuro, a virtualização de redes directamente na infra-estrutura dos operadores de rede irá permitir que os utilizadores interajam e utilizem recursos de rede de uma forma similar ao que fazem com os recursos de CC. O problema é que esta tecnologia ainda está numa fase embrionária e ainda demorará algum tempo até que se torne uma realidade. Se a longo prazo a virtualização de rede deve ser vista como um objectivo a atingir, a curto prazo deverá ser feito um esforço no sentido de trazer algumas das qualidades referidas para as tecnologias de implementação de redes privadas já difundidas entre os operadores de rede.

Actualmente as Virtual Private Networks (VPNs) são o mecanismo mais utilizado pelos utilizadores para integrar os recursos de CC com a sua própria infra-estrutura. No entanto, existem limitações porque esta integração é processada de uma forma estática em escalas de tempo muito superiores aos de CC e na maioria das vezes obrigando os utilizadores a ter que interagir com uma diversidade de operadores.

O objectivo principal desta Dissertação é desenvolver um protótipo que valide o conceito de Cloud Networking utilizando interfaces e protocolos bem definidos, como é o caso da Open Cloud Networking Interface (OCNI) e Open Cloud Computing Interface (OCCI). Este protótipo irá ser capaz de fornecer um serviço integrado de recursos de rede e CC em que o utilizador só irá ter que interagir com um único fornecedor. Isto obriga a que uma linha de comunicação seja estabelecida entre diferentes operadores para uma rápida e automática integração dos dois domínios.

Nesta dissertação irão ser apresentados e estudados interfaces e protocolos que facilitem a interoperabilidade entre operadores com vista à sua implementação num protótipo. No domínio da rede irão ser desenvolvidas ferramentas que permitam o aprovisionamento de recursos com vista a integração de CC com a rede do operador. Irá também ser desenvolvido um orquestrador que permita a um utilizador, através de um único pedido, criar uma infra-estrutura que integre recursos de cloud e rede. Posteriormente irão ser efetuados testes de desempenho da instanciação de um serviço integrado de CC com a rede. Ainda no âmbito desta dissertação irá ser estudada uma plataforma de virtualização de rede na qual irão ser feitos testes de tráfego sobre a mesma com vista à sua integração no protótipo.

Abstract

Cloud Computing (CC) has recently been a very popular theme in the world of technology, its potential is clear both in economic terms and resource efficiency. Despite these qualities, mass adoption of this paradigm is still conditioned by aspects of interoperability, security and QoS. To overcome these aspects, the concept of Cloud Networking arises, which is the integration of CC characteristics in the network.

Currently the network infrastructure is based on technologies that were designed many years ago, at a time when the context and the needs of the world of communications were completely different from today's requirements. This phenomenon has been extensively discussed and referred to as network ossification. This infrastructure is now unable to meet the needs of CC, features as dynamic and flexibility are not reflected in the network. However, the network and CC cannot be separated, because it is the network that provides users the access to these resources and negatively influencing their use.

In the future, the use of network virtualization directly into the network infrastructure of network operators will allow the users to interact and use network resources in a manner similar to what they do with CC resources. The problem is that this technology is still in its infancy and it will take some time until it becomes a reality. If in the long term network virtualization should be seen as an objective, in the short term it should be made an effort to bring some of the qualities referred to the technologies used by network providers to implement private networks.

Currently VPNs are the most used mechanism by users to integrate CC resources with their own infrastructure. However, there are limitations because this integration is processed in a static form on time scales much higher than CC resources, and most often forcing users to have to interact with a variety of operators.

The main objective of this dissertation is to develop a prototype to validate the concept of Cloud Networking using well defined interfaces and protocols, such as OCNI and OCCI. This prototype will be able to provide an integrated service of network and CC resources in which the user will only have to interact with a single supplier. This requires that a communication line is established between different operators for rapid and automatic integration of the two domains.

In this dissertation it will be presented and studied interfaces and protocols that facilitate interoperability between operators with the perspective of implementing them in a prototype. In the network domain, it will be developed tools that allow the provisioning of resources for the integration of CC with operator's network. It will also be developed an orchestrator to allow the user via a single request, to create an infrastructure that includes CC resources and network resources. Later, it will present performance tests of instantiation of integrated services. Also, within the subject of this dissertation, a network virtualization platform will be studied and subjected to performance tests to evaluate its integration on the prototype.

Contents

List of Figures

List of Tables

List of Acronyms

AMQP Advanced Message Queuing Protocol

API Application Programming Interface

ARPA Advanced Research Projects Agency

AWS Amazon Web Services

BGP Border Gateway Protocol

CADF Cloud Auditing Data Federation Working Group

CAPEX Capital Expenditure

CC Cloud Computing

CDMI Cloud Data Management Interface

CE Customer's Edge

CIMI Cloud Infrastructure Management Interface

CloNe Cloud Networking

CMWG Cloud Management Working Group

CoV Coefficient of Variation

CPU Central Processing Unit

CSA Cloud Security Alliance

CTP CloudTrust Protocol

DCP Distributed Control Plane

DMTF Distributed Management Task Force

EC2 Amazon Elastic Cloud Compute

EU European Union

EURO-NF Network of Excellence

FNS Flash Network Slice

FP7 7th Framework Programme for Research and Technological Development

GENI Global Environment for Network Innovations

GEYSERS Generalised Architecture for Dynamic Infrastructure Services

GUI Graphical User Interface

HTML Hyper Text Markup Language

HTTP Hyper Text Transfer Protocol

IaaS Infrastructure as a Service

IEC Internation Electrotechnical Commission

IETF Internet Engineering Task Force

IP Internet Protocol

IPsec Internet Protocol Security

ISO International Organization for Standardization

ISR Infrastructure Service Request

ISV Independent Software Vendor

IT Information Technology

JSON JavaScript Object Notation

MEF Metro Ethernet Forum

MPLS Multiprotocol Label Switching

NA Network Activator

NaaS Network as a Service

NIST National Insitute of Standards and Technology

NM Network Manager

NREN National Research and Education Network

NS Network Slice

NVSS Network Virtualization System Suite

OCCI Open Cloud Computing Interface

OCNI Open Cloud Networking Interface

OGF Open Grid Forum

OS Operating System

OPEX Operational Expenditure

OSPF Open Shortest Path First

OVF Open Virtualization Format

PaaS Platform as a Service

PE Provider's Edge

PT Portugal Telecom

PTIN Portugal Telecom Inovação

pyOCNI Python Open Cloud Networking Interface

QoE Quality of Experience

QoS Quality of Service

RAM Random Access Memory

RIP Routing Information Protocol

ROA Resource Oriented Architecture

SaaS Software as a Service

SAIL Scalable and Adaptive Internet Solutions

SDN Software Defined Network

SLA Service Level Agreement

SME Small and Medium Enterprise

SNIA Storage Networking Industry Association

SOAP Simple Object Access Protocol

SQL Structured Query Language

S3 Amazon Simple Storage Service

TCP Transmission Control Protocol

TRILL Transparent Interconnection of Lots of Links

UDP User Datagram Protocol

URL Uniform Resource Locator

VDC Virtual Data Center

VLAN Virtual Local Area Network

VM Virtual Machine

VPC Virtual Private Cloud

VPN Virtual Private Network

VR Virtual Router

VRF VPN routing and forwarding table

VXDL Virtual private eXecution infrastructure Description Language

VxLAN Virtual eXtensible Local Area Network

WAN Wide Area Network

WSGI Web Server Gateway Interface

W3C World Wide Web Consortium

XaaS X as a Service

XML Extensible Markup Language

4WARD Architecture and Design for the Future Internet

Chapter 1

Introduction

1.1 History of Cloud Computing

The history of CC can be presented in two parts: the first one refers to the birth of the concepts behind CC; the second one refers to the appearance of the first implementations of CC. Between these two there is a gap of approximately thirty years with the first part happening in the 1950s and 60s, when a lot of network related concepts were flourishing but, due to the technical limitations at that time, a lot of them came to a halt.

In the 1950s a computer scientist named Herb Grosch postulated that the entire world would operate on dumb terminals powered by about 15 large clusters of computers. He became famous for Grosch's Law in which he states that the "Computer performance increases as the square root of cost". Although this law has been disproved, some scholars have recently rehabilitated Grosch's law, claiming that "he was correct in his assumption that significant economies of scale and efficiencies could be achieved by relying on massive, centralized data centers rather than an over-reliance on storage in end units" [1].

Later in 1961 John McCarthy was the first to publicly suggest in a speech that computer time-sharing technology might lead to a future in which computing power, and even specific applications, could be sold through the utility business model, like water or electricity [2].

In August 1962 J.C.R. Licklider wrote a series of memos discussing his concept of an "Intergalactic Computer Network" before he came to be a part of the Advanced Research Projects Agency (ARPA) where the first packet switching network was born, which is considered the origin of internet. In those memos Licklider envisioned a globally interconnected set of computers through which everyone could quickly access data and software that would exist on a network, and would migrate to wherever it was needed, which not only describes todays internet but also the basic principle of CC [3]. But it was in 1966 that a canadian researcher and government official named Douglas Parkhill, who, in his 1966 work "The challenge of computer utility" first laid out the elements of the cloud: the comparison to electric utilities and the illusion of infinite and elastic supply [4].

As we can see, the fundamental ideas in CC were established 50 years ago: economy of scale, computing as an utility service, ubiquitous resources and the illusion of infinite and elastic resources. The first time the term Cloud Computing appeared was in 1997, when an information systems professor Ramnath Chellappa gave a lecture in Dallas called "Intermediaries in Cloud Computing : A New Computing Paradigm" [5]. Two years later a company pioneered the concept of delivering enterprise applications via a simple website.

This company was Salesforce, which was founded in 1999 by former Oracle executive Marc Benioff, Parker Harris, Dave Moellenhoff and Frank Dominguez as a company specializing in Software as a Service, SaaS [6].

Around 2004 Amazon realized they were only using a small percentage of their IT infrastructure and they wanted to profit with the remaining. At that time Ben Black, an Amazon website engineer wrote a paper that became notorious for summarizing the idea of using CC to profit from their IT resources. In 2006 Amazon launched Amazon Elastic Cloud Compute (EC2) which is currently the benchmark of CC. Also in this year Google and other companies started to offer browser-based enterprise applications through services like Google Apps [7].

1.2 Current Context

Although CC is one of the top trends in IT world, the industry in general is still offering some resistance to its adoption, which is caused by some important questions that remain to be answered in CC. One of the limitations present in CC is the lack of standards, which hinders providers interoperability and raises some questions within customers about the easiness of moving between cloud providers and migrating their private infrastructures to the cloud. With better interoperability between providers virtual networks can be used to design cloud infrastructures composed by computing, storage and network resources independently of the provider the resources are based on. With this in mind several organizations and associations supported by operators, vendors and government entities have been pushing towards the creation of standards that facilitate the adoption of CC. The problem in this area is the competition between different initiatives covering the same aspects within CC; also the presence in the market of some offers with their own interfaces and protocols have become an obstacle to the introduction and acceptance of these standards.

In 2012 Cisco surveyed more than 1300 IT professionals to find their primary challenges and priorities when migrating their applications and services to the cloud. A cloud-ready Wide Area Network (WAN) was cited as the top priority before moving applications to the cloud, while data security, reliability and control were considered the top obstacles. The time necessary to migrate their infrastructures was also considered a big obstacle, with many of the respondents revealing a complete lack of knowledge over the necessary steps to perform the migration [8]. Some of these questions can be answered with the integration of Cloud Networking and CC; this integration will bring an improvement on security, reliability and performance with a greater separation of the private network from the underlying infrastructure.

1.3 Purpose

The work performed in this Dissertation aims to develop a prototype where an integrated offer of CC resources and network resources can be requested (figure 1.1). First, a network manager will be developed with the purpose of integrating the various interfaces and protocols within the network domain while being responsible for the configuration of the underlying network infrastructure. This tool will have a database where it will keep an updated information on the network resources and save all the information regarding the creation of VPNs.

Various communication platforms will be studied in order to establish connections with the various entities.

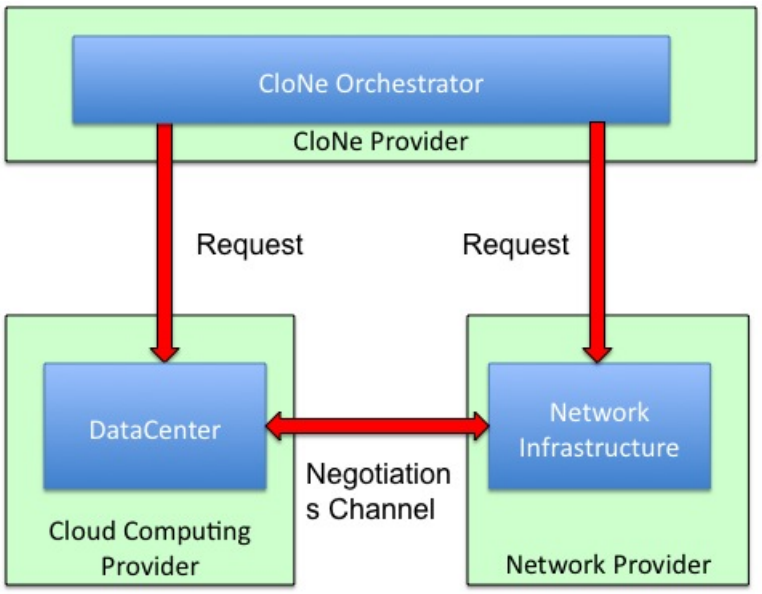


Figure 1.1: Prototype Overview

Secondly, in the CC domain a module will have to be created to work with OpenNebula, an open source software to manage virtualization in data centers. The OpenNebula platform will be studied in detail with special detail to its data base structure and its Application Programming Interfaces (APIs).

Thirdly, a Cloud Networking Orchestrator will be developed capable of accepting requests and instantiate them by using the appropriate interface. To comply with these requirements the different APIs will be investigated and also some of web services will be used in order to deploy the Orchestrator.

1.4 Contribution

As a result of the work performed in this Dissertation, a prototype based on the Cloud Networking (CloNe) architecture defined in the project Scalable and Adaptive Internet Solutions (SAIL) [9] is presented. This prototype is capable of deploying integrated CC and Network resources using open source interfaces and protocols.

The CloNe Orchestrator constitutes the integration element by delegating to lower entities parts of a single request; it uses non proprietary interfaces to communicate with other providers. The Network Manager module provides an integration between the various interfaces and protocols in the network domain and exposes the configuration of VPNs alike services through the use of an API.

The modular approach of the prototype enables multi provider CloNe services deployment in parallel with single provider CloNe services, multi data center or multi network operator domain.

1.5 Dissertation Outline

Chapter 2 gives a definition of what is CC and presents the most important CC providers. A description on the principal associations and organizations that are currently working to provide interfaces and standards to facilitate the adoption by the industry of cloud computing is given. The current top tools and technologies related to CC are overviewed, and it is given a short description on the ongoing initiatives related to this Dissertation.

In chapter 3 a detailed view is provided on the architecture used to implement the prototype. It contains all the parts that compose this work including all their roles within the prototype and their requirements; also, a detailed description on the communication interfaces and protocols used is provided.

The following chapter, chapter 4, describes all the work done in this Dissertation explaining the reasons behind all the steps taken. The main components are deconstructed exposing its main data structures, internal organization, mechanisms and the implementation of the communications platforms used to exchange data between all the modules.

In chapter 5 experimental results are shown and discussed. These results prove the viability of CloNe by showing that all the functionalities work as expected. Also shown, are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) tests performed on the Network Virtualization System Suite to evaluate the possibility of integrating this platform on the prototype.

This Dissertation will end on chapter 6 with a critical review of all the work done in this Dissertation and the obtained results.

Chapter 2

State of the Art

2.1 Introduction

In this chapter, a short view on current technologies and trends regarding the subject of this Dissertation will be provided. It will start by describing CC, its characteristics, service models and deployment models. An overview of some of the biggest cloud providers today is given, with special attention to Amazon, as the reference provider, to AT&T, one of the operators with the most complete cloud offer, and to Portugal Telecom as the national reference provider.

Further, it is pointed out the work of the standardization bodies more active on CC as well as the top Infrastructure as a Service (IaaS) platforms currently available. A brief analysis on the top platforms and technologies used to provide virtualization on data centers and networks will also be given. Finally, it will be provided a short description on current initiatives that address direct or indirectly the same subject of this work.

2.2 Cloud Computing

One of the most popular descriptions of CC belongs to National Insitute of Standards and Technology (NIST), that states "Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"[10]. Although there is no standard definition for CC, the key principles behind it are well known: X as a Service (XaaS), scalability, dynamicity, on-demand self-provisioning and pay-as-you-go. Its economical advantages are directly associated with these principles: CC allows the transfer of Capital Expenditure (CAPEX) into Operational Expenditure (OPEX) by decreasing upfront costs and paying for utility computing [11], e.g. networks or servers, as a service utility just like electricity. CC allows instant and ubiquitous access to resources without the need to over provision to deal with usage spikes, allowing the benefit of only paying for the actual consumption avoiding needless expenditure. Ideally, it should be transparent to the client the required operations by the providers to deploy the services, providing an abstraction of the physical infrastructure.

CC has many benefits; however there are also some associated risks and questions to be an-

swered with using CC; for example, most of the responsibilities are left for the provider which is the one who controls the physical infrastructure where the clients resources are located. Some questions about the capacity to provide security and disaster recovery still remain to be answered. Also the lack of standards limits the interoperability between providers and raises the fear of provider lock-in. It is also important to highlight that most CC offers do not offer end-to-end Service Level Agreements (SLAs), since most rely on the Internet to deliver services. Even if a network service is provided, it is usually provided as a decoupled service, therefore there is no "integrated" SLA.

2.2.1 Characteristics

In CC a client should be capable to configure and scale its resources as he sees fit, and in some cases it should be done automatically without the need to interact with the respective service provider. This provision of resources should be done rapidly and accordingly to the clients demands. The user should have an illusion of infinite resources at his disposal, even though there is always a threshold associated to the physical resources of the provider. Billing must take into account only the client's resource consumption and is usually done in a pay-as-you-go model.

Resource pooling must use the multi-tenancy principle, allowing cost savings through economies of scale, with careful consideration regarding security and data protection. Finally, even though the resources have to be available everywhere, the cloud resources should be independent of the location of physical devices. Their placement should be made taking into account consumer needs while being able to specify location with a higher level of abstraction, e.g. country or region.

2.2.2 Service Models

Traditional CC provides three service models separated by layers of physical resources abstraction, IaaS, Platform as a Service (PaaS) and Software as a Service (SaaS),see figure 2.1. Recently the Network as a Service (NaaS) model has been pointed out as essential to complete the current offers.

- **Software a Service**

  Applications running on cloud infrastructures that can be accessed by consumers with the use of web-browsers or a program interface, leaving to the providers the responsibility of controlling and managing the servers or operating systems where the applications are deployed.

- **Platform as a Service**

  Providers give the clients the ability to deploy applications using the languages, libraries services and tools provided, abstracting the underlying infrastructure,e.g. servers, network or operating systems.

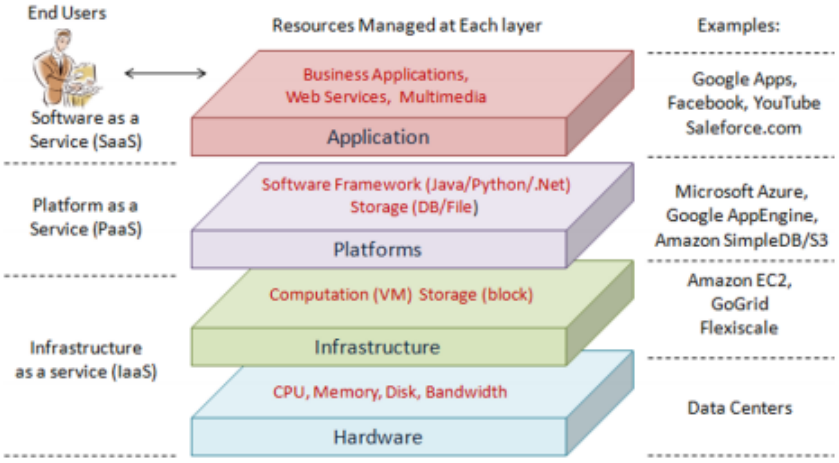


Figure 2.1: Classic layer view of CC [12]

- **Infrastructure as a Service**

  The client can deploy single or clusters of virtualized hardware resources, e.g. virtual machines or storages, ideally with full control and management of the cloud resource. Usually limited configuration over the network architecture is also provided, although the network itself can't be considered as resource.

- **Network as a Service**

  The client can request for a network topology using virtual nodes and links to connect Information Technology (IT) resources, wether they are virtual or real physical resources.

2.2.3 Deployment Models

CC infrastructures can be deployed using different models which can be classified regarding public availability, see figure 2.2.

- Private Clouds are provisioned for exclusive use by a group of consumers regardless of the ownership of physical infrastructures

- Public Clouds are available to the general public although management and control access is owned by a single group

- Hybrid Clouds is an infrastructure that combines Private clouds with Public clouds

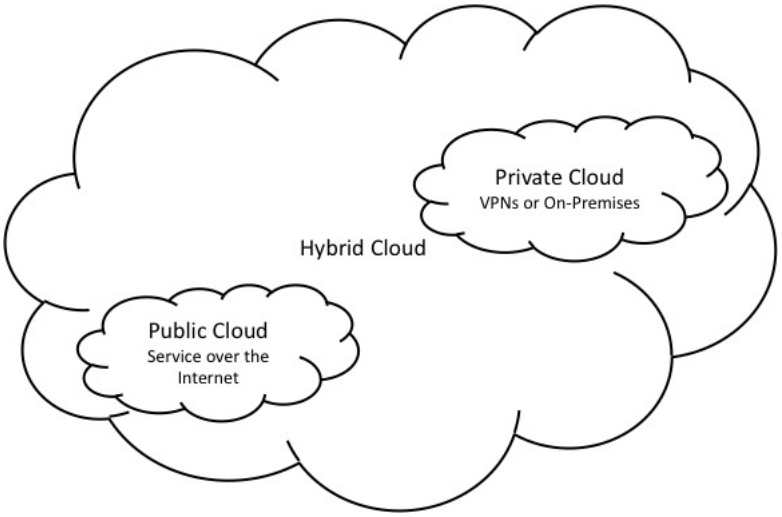


Figure 2.2: CC deployment models

2.3 Cloud Providers

Cloud providers are companies responsible for the delivery of cloud resources regardless of the service models. The number of cloud providers has been growing steadily since Amazon Web Services (AWS) was launched, and today there is already a considerable number of cloud providers on the marker like Rackspace, Google, HP, IBM, AT&T, Amazon and many more. Although there are similarities between the offerings and business models there are some that stand out, like Amazon which is the marker leader and AT&T for having one of the most complete service offers among network operators. In this Dissertation only Amazon, AT&T and Portugal Telecom will be overviewed as cloud providers (the first two because of the reasons mentioned before and the last one because it is the reference within the portuguese market).

2.3.1 Amazon Web Services

AWS is currently the benchmark of CC providers. Estimates claim that it owns around half of the CC market [13] and its EC2 API is considered the standard de facto in IaaS cloud model. AWS main CC related offers are currently centered around four services: EC2, Amazon Simple Storage Service (S3), Virtual Private Cloud (VPC) and DirectConnect, which can be associated with each other. EC2 provides a elastic and flexible IaaS in which the client can configure instances in terms of Central Processing Unit (CPU), Random Access Memory (RAM), Operating System (OS) and software packages. Also, at the client disposal is the ability to reconfigure the instances within minutes with automatic scaling as an option. Within the IaaS cloud model, AWS offers a scalable, flexible, reliable and secure cloud storage through S3, which can be used to distribute and store content. Regarding NaaS, AWS has two products: VPC and DirectConnect. VPC allows the client to build a virtual network for the EC2 resources with the option to configure private and public subnets for one or all of the instances; it also allows to connect the VPC to the client external infrastructures by using

an Internet Protocol Security (IPsec) VPN. DirectConnect is a recent product within the AWS in which the client can connect to any of the previous mentioned services using network providers based VPNs. This product is only available in some of the Amazon data centers, three in North America, two in Asia and one in Europe. It should be noted that no SLAs are available for VPC and DirectConnect [7].

2.3.2 AT&T

AT&T has recently announced the release of "AT&T Synaptic Compute as a Service" in North America, by combining their own VPN technology with VMware cloud infrastructure software. Besides this service they offer cloud storage with "AT&T Synaptic Storage" and "AT&T Platform as a Service". Their PaaS service offers a platform in which Independent Software Vendors (ISVs) can develop and deploy web applications with a redundant and scalable infrastructure running in the background; they also provide various customizable templates for software development. With "AT&T Synaptic Compute as a Service", a client can create virtual data centers with individual Virtual Local Area Network (VLAN), IP addresses and flexible resource sizing similar to other cloud providers. Being a network service provider, they can offer Multiprotocol Label Switching (MPLS) based VPNs, besides the more traditional IPsec, between the Virtual Data Centers (VDCs) and the clients private networks, with SLAs [14].

2.3.3 SmartCloudPT

Portugal Telecom's current offers in the area of cloud computing is composed of two models, IaaS and SaaS. Under the service "Servidores Virtuais" they provide private and public scalable virtual data centers with limitless number of servers with customizable topology and integration with external infrastructures, e.g. the client own data center. Since Portugal Telecom is also a network provider, there is the ability to integrate the virtual data center in the client private network with guaranteed bandwidth to the VPN or the internet. "Desktop Remoto" gives the client access to a virtual machine with three standard configuration options in terms of CPU, RAM and storage capacity, with guaranteed bandwidth access to a VPN or the internet. "Drive Virtual" provides cloud storage similar to Dropbox while "Pack Microsoft Office 365" has Email, calendar and Office web apps, but more directed to enterprise clients. "PHC Business FX" is a management software for micro and small companies allowing to manage clients, suppliers, stocks and also income and expense accounts [15].

2.4 Cloud Standardization Bodies

Before Cloud computing can become a real world alternative to traditional IT resources common standards should be defined to improve cloud providers interoperability and facilitate the adoption of CC. Only recently some steps have been taken towards the definition of standards in this domain, but as soon as major players started taking interest it became more of race between numerous organizations backed up by CC providers and companies all around the world, even those that are not directly related to CC. For now, given his position

as CC leader, Amazon's API has become the reference and it is being adopted by many CC providers and platforms even though it is not really a standard.

There are a number of organizations that ratify proposals for open standards and guidelines for CC. In spite of the latter being the central theme, they end up approaching different points. For example, NIST is approaching CC in a transverse manner with a primary objective of easing the adoption of CC by the industry while safeguarding their economic activities. Open Grid Forum (OGF), Distributed Management Task Force (DMTF) and Storage Networking Industry Association (SNIA) have developed an important work in the area of APIs for infrastructures, Cloud Security Alliance (CSA) has worked more in the area of security, and Metro Ethernet Forum (MEF) and IETF are addressing network related issues.

2.4.1 NIST - National Institute of Standards and Technology

NIST [16] is a non-regulatory agency of the United States Department of Commerce responsible for promoting innovation and industrial competitiveness by advancing measurement science, standards and technology to enhance economic security. NIST area of focus is technology, and specifically, interoperability, portability and security requirements, standards and guidance to promote a secure and effective adoption of the cloud computing model. NIST has five working groups in distinct areas:

- Reference Architecture and Taxonomy
- Standards Acceleration to Jumpstart the Adoption of Cloud Computing
- Cloud Security
- Standards Roadmap
- Business Use Cases

2.4.2 DMTF - Distributed Management Task Force

DMTF [17] is an organization supported by some of the biggest players in IT to promote standards for systems management to improve interoperability between IT products from different companies. They have released Open Virtualization Format (OVF) which, according to them, describes an "open, secure, portable, efficient and extensible format for the packaging and distribution of software to be run in virtual machines" that is hypervisor and processor architecture independent. They have two working groups regarding cloud standards development:

- Cloud Management Working Group (CMWG) which develops prescriptive specifications that deliver architectural semantics as well as implementation details to achieve interoperable management of clouds between service developers and providers. This group has alliances with the other standardization bodies like the CSA, OGF, SNIA and TeleManagement Forum. They are currently working to deliver a new standard, Cloud Infrastructure Management Interface (CIMI), which defines a logical model for the management of resources within the IaaS domain.

- Cloud Auditing Data Federation Working Group (CADF) will develop an audit event data model and a compatible interaction model that is able to describe interactions between IT resources suitable for cloud deployment models. This group is working with CSA and The Open Group [17].

2.4.3 OGF - Open Grid Forum

The OGF [18] is a community of users, developers and vendors standardization of grid computing. Its work is done in open forums where best practices and tendencies are explored to be consolidated in norms. They have a working group responsible for defining the OCCI which seeks to help IaaS providers to deliver compute, storage and network resources through a standardized interface. Its features are integration, portability, interoperability and extensibility [18]. Its main competitor is the CIMI, the Amazon EC2 API can not be considered a competitor since it is proprietary.

2.4.4 SNIA - Storage Networking Industry Association

SNIA [19] is an association of producers and consumers of storage networking products dedicated to the promotion of standards and activities regarding the storage and management of information. Within the SNIA, the Cloud Storage Initiative is promoting the adoption of cloud storage as a new delivery model and it is currently working in the Cloud Data Management Interface (CDMI), which is an SNIA standard for associating data services with the data itself with the use of tags [19].

With the use of OCCI/CIMI and CDMI, a parallelism can be made with the Amazon APIs, EC2 for infrastructure and S3 for storage, see figure 2.3

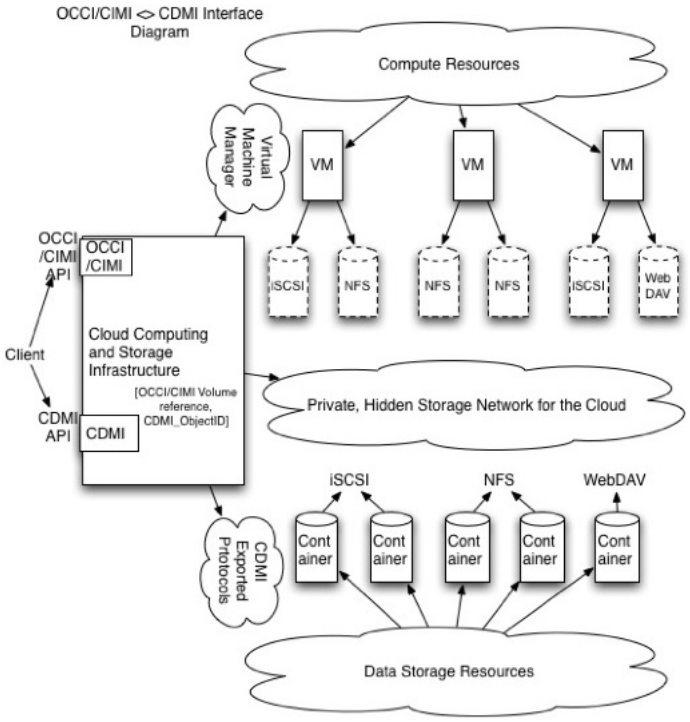


Figure 2.3: CDMI and OCCI/CIMI in an integrated Cloud Computing Environment [20]

2.4.5 CSA - Cloud Security Alliance

The CSA [21] is a member-driven organization promoting the use of best practices for providing security assurance within CC. It incorporates some of the big players in cloud computing like Amazon, Microsoft and Rackspace as well Telecommunication companies and Hardware vendors. The CSA has an extensive list of initiatives regarding the security and promotion of CC, some of them are:

- Security Guidance for Critical Areas of Focus in Cloud Computing: Foundational best practices for securing CC
- Big Data Working Group: Working group created to identify best practices for security and privacy in big data
- Cloud Controls Matrix: Security controls framework for cloud providers and cloud consumers
- CloudTrust Protocol (CTP): The CTP is the mechanism by which cloud service consumers ask for and receive information about the elements of transparency as applied to cloud service providers
- Security as a Service: Research for gaining greater understanding for how to deliver security solutions via cloud models

- Top Threats to Cloud Computing: Provide needed context to assist organizations in making educated risk management decisions regarding their cloud adoption strategies
- Cloud Audit: Provide a common interface and namespace that allows CC providers to automate the audit, assertion, assessment and assurance of their IaaS, PaaS and SaaS models
- Cloud Metrics: Metrics designed for Cloud Controls Matrix and CSA Guidance [21]

2.4.6 IETF - Internet Engineering Task Force

The Internet Engineering Task Force (IETF) [22] is an international community of network operators, vendors and investigators which develop and promote internet standards cooperating closely with World Wide Web Consortium (W3C) and International Organization for Standardization (ISO)/Internation Electrotechnical Commission (IEC) standards bodies. They are organized into working groups and informal discussing groups and, although none of them is currently working in CC, some address directly or indirectly the subject. The standard Transparent Interconnection of Lots of Links (TRILL) and the project Virtual eXtensible Local Area Network (VxLAN) tackle the issue of excess growth in numbers of Virtual Machines (VMs) and VLANs inside data centers due to CC. There is also the project "Requirements and Framework for VPN-Oriented Cloud Services" and a discussion group with the theme "Discussion of issues associated with large amount of virtual machines being introduced in data centers and virtual hosts introduced by Cloud Computing"[22].

2.4.7 MEF - Metro Ethernet Forum

The MEF [23] is a nonprofit international industry consortium, dedicated to worldwide adoption of ethernet networks and services. The forum is composed by more than 175 organizations, including network operators, vendors and semiconductors suppliers. In 2011 the MEF started a project focused on the role of ethernet networks and services in the delivery of Private Clouds[24], which has released so far a white paper called "Carrier Ethernet for Delivery of Private Cloud Services" where it states the importance of WAN for improving the Quality of Experience (QoE) in Cloud services[23].

2.5 Cloud Management Platforms

Cloud Management Platforms are software suites for creating, managing, monitoring and deploying infrastructure cloud services. They use hypervisors to instantiate and manage virtual resources and provide APIs to allow users to control and manage virtual resources. Due to the development of CC, many platforms have emerged in recent years, some have proprietary licenses like Eucalyptus and VMware vCloud while others are open-source like OpenNebula, CloudStack and OpenStack. A detailed overview will be given on OpenStack, which recently, due to the larger support by major players in the market has become the platform of reference. OpenNebula was the one used in this Dissertation, due to the fact that, in the beginning of this work, OpenNebula had a larger support and offered a more

stable release while fulfilling all the requirements.

2.5.1 OpenStack

OpenStack [25] is an open source software for cloud providers and customers which enables the creation of Computing, Network and Storage resources on standard hardware. It was initially a joint venture between Rackspace Cloud and NASA but currently around 178 company's are also participating in his development, including major players from different areas like Cisco, IBM or HP. OpenStack uses a modular approach which is divided in four products:

- Compute: provision and manage large networks of virtual machines
- Storage: object and block storage for use with servers and applications
- Networking: pluggable, scalable, API-driven network and IP management
- Dashboard: web app that allows the control of computing, storage and networking resources

OpenStack supports Amazon EC2 API and also a proprietary API in common with Rackspace Cloud.

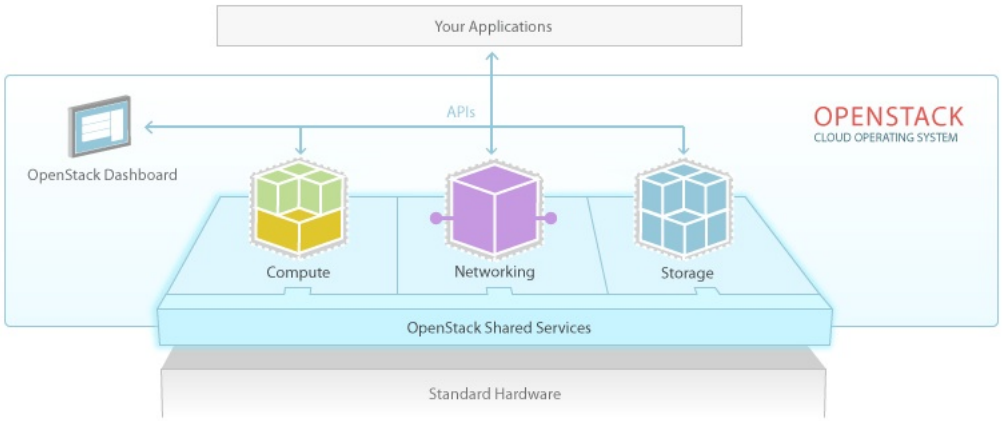


Figure 2.4: OpenStack Overview [25]

2.5.2 OpenNebula

OpenNebula [26] is an open source software to manage virtual infrastructures on data centers. Although basic functionality is similar to OpenStack, one of the differentiating feature is the ability to deploy hybrid clouds completely transparent for the user. In case the private infrastructure is having a resource shortage, some instances may be deployed in external data centers like Amazon.

OpenNebula, unlike OpenStack, uses an integrated approach in which Storage and Network resources are always associated with a Compute resource, although both of them are configured individually. Various hypervisors are supported with the ability to control and monitor virtual machines. The main hypervisors are Xen, Kvm and VMware. To interact with OpenNebula several interfaces are provided, which allow the management of the virtual machines. The main interfaces are the command-line, Sunstone Graphical User Interface (GUI), OCCI and EC2 .

Figure 2.5: OpenNebula Overview [26]

2.6 Network Slicing

Within the cloud networking context, a Network Slice (NS) represents a virtual network isolated from the physical network. The term slice derives from the fact that each virtual network is seen as a separate network infrastructure operating on top of the physical network, see figure 2.6 [27].

The development of CC caused an increase in terms of customer requirements on network operators. The great advantages of CC, like scalability and flexibility, must not be constrained by network limitations. Currently, network operated VPNssare the most used technique to deploy virtualized networks due to its maturity, robustness and reliability. The downside is that VPNs were not designed to cope with CC characteristics like elasticity, on-demand reconfiguration and resource mobility. Normally a VPN has a relatively long life-cycle compared to a CC resource, e.g. virtual machine, and normally there are no changes to its configuration during its lifespan.

Forced by the increasing network requirements, vendors and network providers have been developing new ways to deliver network services.Two approaches have emerged, Software Defined Network (SDN) and Virtual Networks.

2.6.1 Legacy Virtual Private Networks

A VPN [28] is a private data network that uses tunneling protocols and security procedures to provide separation from the public telecommunication infrastructure where it is deployed. The tunneling protocol encapsulates the frame in an additional header. This provides routing information so that the encapsulated payload can traverse the intermediate network. The encapsulated packets are then routed between tunnel endpoints over the Internet. Once the

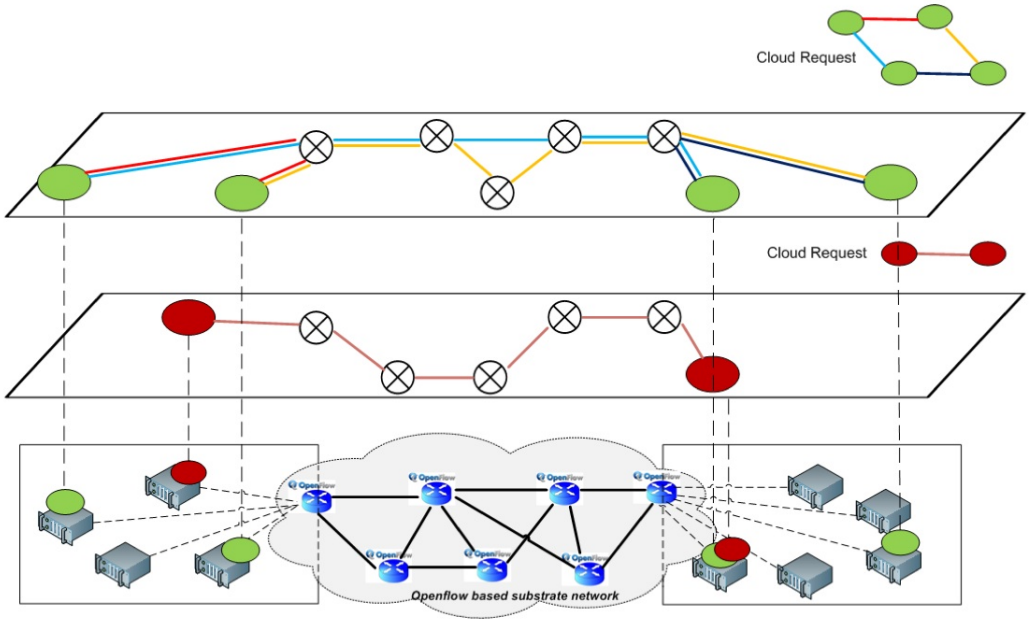


Figure 2.6: Network slicing using OpenFlow [27]

encapsulated packets reach their destination, the frame is decapsulated and forwarded to its final destination. Tunneling includes this entire process (encapsulation, transmission and decapsulation of packets).

VPNs can be deployed using two different models. The first is using the customers private equipments in which the network plays a mere transport role; and the second one in which the configurations, operation and control procedures are provided by network service providers. Although the first one can guarantee limited levels of confidentiality and access control, it is not possible to have full management capabilities in terms of end-to-end performance and reliability which will hinder the adoption of cloud computing by enterprises.

In the second model, the VPN is established between edge routers providing a robust, reliable and secure network path between customer sites and cloud resources. One of the most used configurations is Border Gateway Protocol (BGP)/MPLS. This latter method uses a "peer model", in which the Customer's Edge (CE) routers send their routes to the service Provider's Edge (PE) routers. BGP is then used by the service provider to exchange the routes of a particular VPN among the PE routers that are attached to that VPN. This is done in a way that ensures that routes from different VPNs remain distinct and separate, even if two VPN have an overlapping address space. The PE routers distribute to the CE routers in a particular VPN, the routes from other CE routers in that VPN. Each route within a VPN is assigned a MPLS label. When BGP distributes a VPN route, it also distributes an MPLS label for that route. Before a customer data packet travels across the service provider's backbone, it is encapsulated with the MPLS label that corresponds, in the customer's VPN, to the route that is the best match to the packet's encapsulated address. This MPLS packet is further encapsulated so that it gets tunneled across the backbone to the proper PE router; thus, the backbone core routers do not need to know the VPN routes.

2.6.2 Software Defined Networks

Typical network devices, e.g. routers and switches, concentrate data plane and control plane. SDN proposes to separate this two by moving the control plane to a centralized entity thus allowing to control the entire paths of packets within the network. A parallelism can be made between computers and SDNs: looking at a computer in terms of layers, there is the bottom layer, the hardware, that interacts with the middle layer, the OS, which in turn enables the deployment of applications. In SDNs the same rules apply: the bottom layer, e.g. switches, interact with the middle layer using APIs like OpenFlow[29] which will enable in the future the deployment of network applications [30].

OpenFlow is a communication protocol that allows a controller, also known as the network OS, to program the flow-table in different switches and routers including virtual network devices, see figure 2.7. Several vendors are supporting OpenFlow like Juniper Networks or Cisco. OpenFlow key attributes are the separation of data and control planes, a uniform vendor-agnostic interface between control and data planes, a logically centralized control plane and slicing, and virtualization of the underlying network [29].

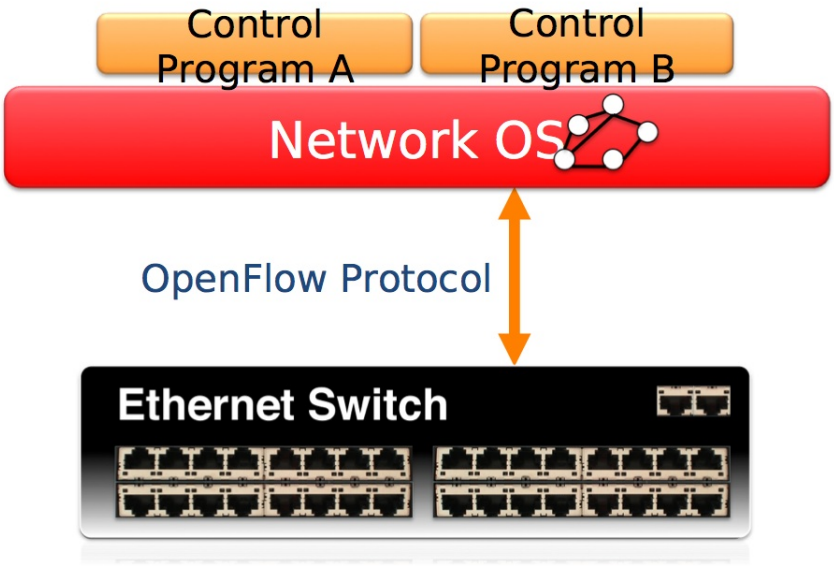


Figure 2.7: OpenFlow Protocol within Software Defined Networks [31]

2.6.3 Network Virtualization

Network virtualization consists in the virtualization of network resources, links and nodes, on top of the physical infrastructure decoupling the virtual network from the underlying infrastructure. Although VPNs and SDNs can be used to deploy virtual networks they are still integrated in the physical substrate and should be seen more as a service rather than a real network. The decoupling of the virtual network from the underlying infrastructure allows

a more flexible allocation and configuration of resources, enabling the deployment of different architectures and protocols using a single infrastructure, see figure 2.8 [27].

Before network virtualization can be seen as a real alternative to more conservative approaches, it still needs to answer to some challenges, like the performance of virtualized network resources and its management and monitoring. Currently it is only available in small-scale testbeds like the Network Virtualization System Suite (NVSS) [32] developed under the Architecture and Design for the Future Internet (4WARD)[33] project and Global Environment for Network Innovations (GENI)[34]. An overview of these platforms is provided bellow, special attention is given to the NVSS due to its importance to this Dissertation.

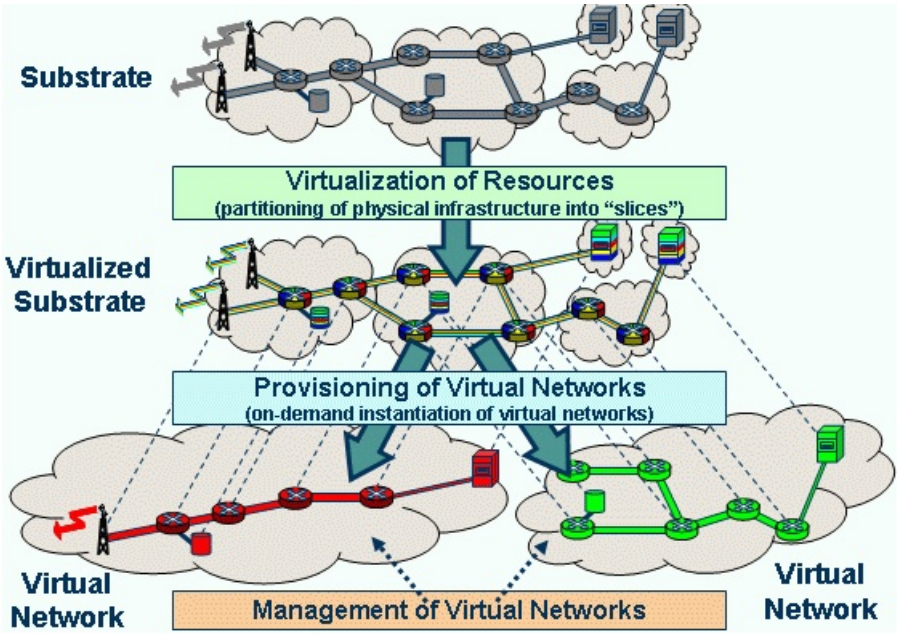


Figure 2.8: Virtual Network layer view [27]

GENI

GENI is an initiative funded by the National Science Foundation in the United States. It includes a large-scale infrastructure to help developing the future internet. Within the infrastructure virtual nodes and links can be deployed to test new and sometimes disruptive network technologies. Due to the virtual nature, this infrastructure can be shared between researchers. GENI has a number of different resources available: backbones resources which are geographically distributed but can be connected using Layer 2 or Layer 3 VLANs, programmable hosts in which VMs can be deployed, programmable switches and also wireless testbeds for experimentations [34].

Network Virtualization System Suite

The Network Virtualization System Suite, NVSS, is a platform for the creation, discovery, monitoring and management of virtual networks which was developed under the 4WARD project [32]. The platform is divided in three modules, the Control Centre which provides a Graphical User Interface for the manager, the Manager which is the central module of the NVSS and the Agents which are present in the physical nodes and allow the Manager to interact with them.

- The Control Centre was developed in Java and uses sockets to communicate with the Manager allowing to create, manage and monitor virtual networks through a drag-and-drop mechanism.
- The Agent module must be present in every physical node within the network. It retrieves information regarding the physical node and forwards it to the Manager, provides discovery functions through a distributed algorithm and executes orders received from the Manager on the node.
- The Manager is the intermediary between the Control Centre and the Agents. It receives orders from the Control Center while providing feedback from the network. It gathers information provided by the Agents allowing to draw the substrate and virtual networks topologies while keeping a database with an up-to-date information about the deployed virtual networks and the physical resources [32].

The Manager and the Agents were developed using C Language. The main features of the NVSS are:

- Physical and Virtual Topology Discovery
- Virtual Network Creation
- Virtual Network Monitoring
- Virtual Network Management

2.7 Ongoing Initiatives

Currently there are several ongoing initiatives under the 7th Framework Programme for Research and Technological Development (FP7) [35]. The main goal of the Framework Program is to strengthen the scientific and technological base of european industry and to encourage its international competitiveness, while promoting research that supports EU policies. Within the FP7 there are three programs that address the same subject, design the basis for the Network of the Future, which this Dissertation relates to.

2.7.1 EURO-NF

Network of Excellence (EURO-NF) [36] is a project formed by 35 institutions from the academia and industry. Its main target is to integrate the research effort of the partners

to be a source of innovation and think tank on possible scientific, technological and socio-economic trajectories towards the network of the future. The project is divided in three research domains:

- The first domain relates to future services an network architectures that have to integrate natively a global mobility of users, services, terminals and networks
- The second research domain is related with the quantitative issues of the networks of the future, such as new modeling, design, and dimensioning tools contributing to the development of a network theory, in particular for the efficient processing of traffic
- Finally, giving the exponentially growing impact of the internet, topics like governance, regulation policies, privacy vs security, standardization will be studied together with the design of the network [36]

2.7.2 GEYSERS

Generalised Architecture for Dynamic Infrastructure Services (GEYSERS) [37] is a project gathering National Research and Education Networks (NRENs), universities, research institutions, Small and Medium Enterprises (SMEs) and manufacturers focused on qualifying optical infrastructures providers and network operators with a new architecture. The project has two goals [37]:

- Specify and develop mechanisms that allow infrastructure providers to partition their resources by creating specific logical infrastructures and offer them as service to network operators.
- Specify and develop a Network Control Plane for the optical Infrastructure which will allow to couple optical network connectivity and IT services automatically, dynamically and on-demand.

2.7.3 SAIL

SAIL [9] is a project which consists of a consortium of operators, vendors and research institutions collaborating to develop the networks of the future. SAIL will both design technologies and develop techniques to transition from today's networks to the networks of the future. SAIL leverages state-of-the art architectures and technologies, extends them as needed, and integrates them. SAIL uses experimentally-driven research, building prototypes that will proof the advantages in concrete use cases. In order to address these challenges, the work is organized in four research areas [9]:

- Migration, Standardization, Business and Socio-Economics - It co-ordinates various cross-cutting themes that affect each of the other research areas and will also ensure that standardization and migration, socio-economics, business and regulatory aspects, and exploitation and dissemination activities are properly considered
- Network of Information - It aims at demonstrating Network of Information reduced overall cost per delivered bit of application payload, for some reasonable and accepted

cost metrics. Finally, it will characterize the scaling properties of important figures of merit of the proposed system

- Open Connectivity Services - It aims at developing and implement a proof-of-concept protocol capable of controlling transport data flows from end-to-end and edge-to-edge over Internet Protocol (IP) and MPLS networks, and new emerging transport technologies, efficiently utilizing diverse routes and transport capabilities. Accordingly, it will design a comprehensive ecosystem and will provide validation by prototypes in a network-level testbed, for managing data transport flows between edge interfaces

- Cloud Networking - It aims at designing a novel cloud networking architecture supporting flash network slices, plus a management and a security framework that will be evaluated through the development of a large scale prototype distributed across different sites in Europe

2.8 Summary

This chapter provided the reader with an overview of CC and its various service and deployment models. In an effort to provide a better contextualization for the work done in this Dissertation, it was highlighted some of the most prominent offers in the market as well as some of the activities of the major standardization bodies in CC. Cloud providers are adopting network operated VPNs to improve their services, but their deployment usually takes several days and sometimes lacking *end2end* SLAs. To ease this process, major standardization bodies have been working in the creation of standards; although it should be noted that their work has resulted in a competition for standardization, which increases divergence.

Furthermore, it was given a description on the cloud management platform used in this work, OpenNebula, and its main competitor, OpenStack. An overview on all the technologies for network slicing was also given. Finally, it was pointed out some of the current initiatives funded by the European Union (EU) that relate to this work.

Chapter 3

Architecture & Mechanisms Design

3.1 Introduction

The aim of this chapter is to give a detailed view on the architecture and the various modules that compose this work. It will start by describing the architecture model used and its various elements, and it will then describe the prototype. Afterwards all the communication interfaces and protocols used to exchange data between the various domains will be thoroughly explained. To finalize this chapter, all the modules that compose the prototype will be described in terms of their roles within the prototype. The modules will be divided by the domain in which they are inserted, starting at the top layer and then moving to the lower layer where the Cloud Computing domain and the Networking domain are positioned.

3.2 Cloud Networking Overview

The CloNe architecture is based on the architecture defined in the deliverable provided by the Work Package D in the SAIL project. It uses a three layer model which is composed by the resource layer, single domain infrastructure layer and cross domain infrastructure layer (figure 3.1). The resource layer includes computing resources, storage resources and network resources as virtual entities that reside within the boundaries of a single administrative domain; they can be created, managed and destroyed. The creation of a virtual resource will correspond to the consumption of a physical resource, e.g. disk space, or a logical resource, e.g. bandwidth; and its destruction implies freeing the resources previously allocated. The single domain infrastructure is a collection of resources managed by a single domain authority, which has full knowledge and management rights over the underlying equipment and virtualization technology. Cross domain infrastructure represents an aggregate of interconnected virtual resources that can be partitioned into multiple single domain infrastructures. Single domain infrastructures and cross domain infrastructures can be created, updated, managed and destroyed.

Limited information will be exchanged between authorities of administrative domains in order to facilitate the optimization of cross domain virtualization. These communications can be divided in two areas: horizontal communication which is established between infrastructure service providers within the same layer; and vertical communication which is established between a single domain infrastructure service provider and a cross domain infrastructure service provider.

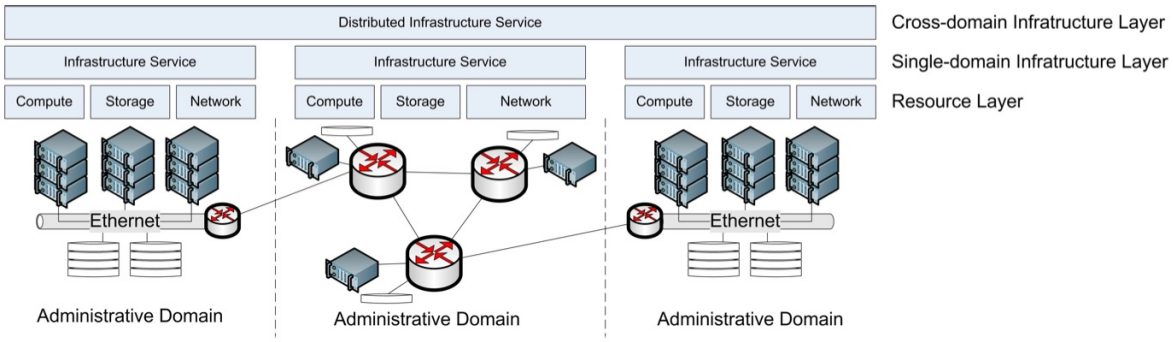


Figure 3.1: CloNe three layer architecture [27]

An user/organization can access a cross domain infrastructure service provider in order to obtain, examine, modify and destroy resources. The creation of an infrastructure is achieved with the user sending an Infrastructure Service Request (ISR) to a service provider, delegating to the latter the responsibility of implementing the virtual infrastructure by partitioning the request into parts that are forwarded to lower providers. This delegation hierarchy among providers will be transparent to the user.

Important Concepts

- **Infrastructure Service Request**

  The ISR is a high level description of functional and non-functional user requirements allowing the user to integrate in a single infrastructure resources that may be largely distributed.

- **Flash Network Site**

  Within the CloNe context an Flash Network Slice (FNS) represents a network resource that provides communication capabilities with associate QoS parameters. It can be attached to other resources through links, e.g. a link can connect a VM to a FNS or two FNSs. Although a link may span administrative domains the FNS itself is created and managed within a single administrative domain.

3.2.1 Proposed Solution

Looking at the figure 3.2 it is possible to identify the architecture previously described. The CloNe provider is the one which receives the request from the infrastructure service user and acts on his behalf in the creation of the infrastructure. Using the appropriate channels it delegates to the lower providers in the Cloud, namely datacenters, and Network domain the creation of the resources, Only the Cloud and Network authorities (represented in the figure by the Cloud Management tools and Network Management Tools) can interact directly with the resources.

To instantiate the network services, two choices were considered in the beginning, the NVSS platform and legacy VPNs. After careful consideration, the choice resides in the legacy VPNs because they are a more faithful representation of a network provider infrastructure in a short-term approach.

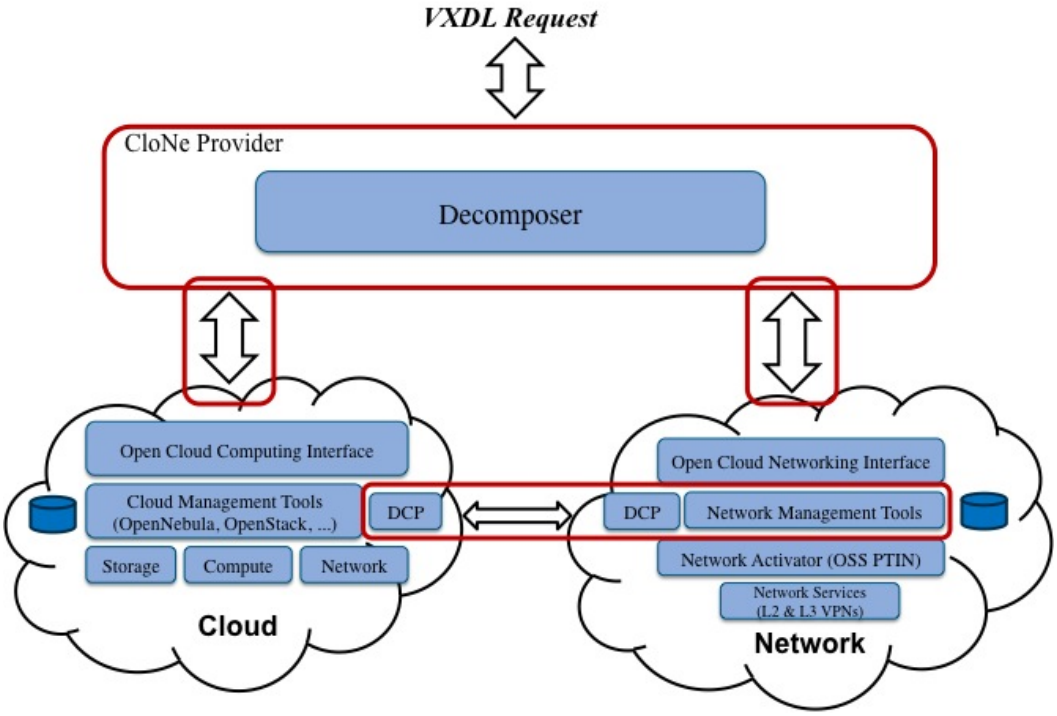


Figure 3.2: CloNe prototype

The red areas identify the main focus of this Dissertation: the integration of the Distributed Control Plane (DCP) in both the Cloud and the Network domain, the development of Network Management Tools, and the CloNe Orchestrator.

3.3 Interlayer Communication

Two different and complementary interfaces will be used to delegate lower level requests between the cross domain provider and single domain providers. In terms of computing resources, which can be computing, storage or network, here network refers to internal data center network resources. The OCCI was chosen because of its maturity and growing adoption in CC, e.g. OpenNebula [26] and OpenStack [25] already have an implementation of OCCI in their platforms. Regarding network resources, e.g. FNS, the OCNI will be used, OCNI is a network centric extension to the OCCI capable of describing connectivity between sites.

3.3.1 Open Cloud Computing Interface

The OCCI is a boundary RESTful Protocol and API that acts as a front-end primarily for IaaS model-based services. Due to his extensibility, it also serves other models including

PaaS, SaaS and NaaS. It was developed to provide a common interface for various tasks, such as deployment, autonomic scaling and monitoring.
Due to the modular nature of the OCCI the complete specification was separated in three documents:

- OCCI Core Model: specification of the OCCI Core Model which can be interacted with renderings and expanded through extensions
- OCCI Rendering: specifications describing the rendering of the OCCI Core Model and can be extended following the rules in the specification
- OCCI Extensions: specifications describing extensions to the OCCI Core Model.

Core Specification

The OCCI Core Model consists on the definition of its four base types:

- Entity- The Entity type is an abstraction of the Resource and Link type and as such it cannot be instantiated. Entity enforces for all sub-types a required occi.core.id attibute, unique identifier of the resource instance, and an optional occi.core.title attribute, a name assigned to the resource instance. Every sub-type of Entity, also referred as resource instance, must be assigned to a Kind instance, which represents a unique type identifier for a particular sub-type of Entity. The Mixin type is a complement to a resource instance allowing to extend their capabilities representing an extension mechanism to the OCCI Core Model and, because of that, it is an optional attribute. All the attributes defined for the Entity type can be seen in table 3.1.
- Resource- The Resource type inherits Entity and represents a general object which can be related to real-world resource like a virtual machine or a network service. It can have an occi.core.summary, which is a description of the Resource instance and it can be associated with another resource through a link. All the attributes defined for the Resource type can be seen in table 3.2.
- Link- A Link type establishes a relation between Resource instances and it also inherits Entity. Its attributes are occi.core.source and occi.core.target which indicate the Resource instances connected. A Link may point to an external resource thus providing a mean to extend the OCCI Core Model. All the attributes defined for the Link type can be seen in table 3.3.
- Action- The Action type is an abstract type which defines an invocable operation to a Resource instance or a collection, which are groups of Resources that are associated with either the same Kind or Mixin. An Action relates to a capability of either a Kind or a Mixin instance and it can be invoked on any Resource instance associated with the same Kind or Mixin. All the attributes defined for the Action type can be seen in table 3.4.

To extend OCCI, the OCCI Core Model has three methods available [38]:

- adding Category, Kind and Mixin instances

Attribute	Type	Multiplicity	Description
occi.core.id	URI	1	A unique identifier of the Entity sub-type instance
occi.core.title	String	0..1	The display name of the instance
kind	Kind	1	The Kind instance uniquely identifying the Entity sub-type of the resource instance
mixins	Kind	0..*	The Mixin instances associated to this resource instance

Table 3.1: **Entity type Attributes** - Attributes defined for the Entity type [38]

Attribute	Type	Multiplicity	Description
occi.core.summary	String	0..1	A summarizing description of the Resource instance
links	Link	0..*	A set of Link compositions

Table 3.2: **Resource type Attributes** - Attributes defined for the Resource type [38]

Attribute	Type	Multiplicity	Description
occi.core.source	Resource	1	The Resource instance the Link instance originates from
occi.core.target	Resource	1	The Resource instance the Link instance points to

Table 3.3: **Link type Attributes** - Attributes defined for the Link type [38]

Attribute	Type	Multiplicity	Description
category	Category	1	The identifying Category of the Action, it associates the Action type with the Kind and Mixin instances

Table 3.4: **Action type Attributes** - Attributes defined for the Action type [38]

- sub-typing by adding Resources, Links and Actions
- using Mixins which can be associated with any Resource

OCCI Rendering

OCCI Rendering is built upon Resource Oriented Architecture (ROA) and uses many of the features of Hyper Text Transfer Protocol (HTTP) and underlying protocols. The use of HTTP provides the means to uniquely identify resources through Uniform Resource Locators (URLs), while allowing the use of HTTP verbs to operate upon them, Create (POST), Retrieve (GET), Update (POST/PUT) and Delete (DELETE). Table A.2 contains the HTTP return codes and its description defined in the OCCI Rendering, while table 3.5 contains the actions defined for the HTTP verbs which vary depending on the object pointed by the URL.

Path	GET	POST	POST (Action Query)	PUT	DELETE
resource instance	Retrieval of the resource instance's representation	Partial update of the resource instance	Perform an action	Creation Update of the resource instance, supplying the full representation of the resource instance	Deletion of the resource instance
Kind collection	Retrieve a collection of resource instances	Create a new resource instance of this kind	Performs actions on a collection of resource instances	Not defined	Removal of a single, a subset or all the resource instances from the kind collection
Mixin collection	Retrieve a collection of resource instances	Adds a resource instance to this collection	Performs actions on a collection of resource instances	Update of the collection supplying the full representation of it. Includes removal and addition resources	Removal of a single, a subset of or all the resources instances from the Mixin collection
query interface	Retrieve capabilities	Add a user-defined Mixin	Not Defined	Not Defined	Removal of a user-defined Mixin

Table 3.5: **HTTP Verbs** - HTTP verb behavior summary [39]

OCCI Infrastructure

An OCCI implementation can model and implement an API for a OCCI service model, in this case IaaS, allowing for the creation and management of resources. The main infrastructure types defined within the OCCI infrastructure are:

- Compute- Information processing resources, e.g. a virtual machine. Start, stop, restart and suspend are the Actions defined for Compute by its Kind instance. All the attributes defined for the Compute type can be seen in table 3.6.
- Network- Interconnection resource, although it represents a L2 networking resource, it can be extended to support L3 and L4 capabilities using the mixin mechanism. Up and

down are the only Actions defined for the Network by its Kind instance, allowing to set active or inactive state. All the attributes defined for the Network type can be seen in table 3.7.

- Storage- Information recording resources. Online, offline, backup, snapshot and resize are the Actions defined for Storage by its Kind instance. OCCI can be used in conjunction with the SNIA cloud storage standard, and CDMI to enhance the management of the cloud computing storage and data. All the attributes defined for the Storage type can be seen in table 3.8.

Attribute	Type	Multiplicity	Description
occi.compute.architecture	Enum	0..1	CPU Architecture of the instance
occi.compute.cores	Integer	0..1	Number of CPU cores assigned to the instance
occi.compute.hostname	String	0..1	Fully Qualified DNS hostname for the instance
occi.compute.speed	Float	0..1	CPU clock frequency in gigahertz
occi.compute.memory	Float	0..1	Maximum RAM in gigabytes allocated to the instance
occi.compute.state	Enum	1	Current state of the instance. Possible states are active, inactive and suspended

Table 3.6: **Compute Attributes** - Attributes defined for the Compute type [40]

Attribute	Type	Multiplicity	Description
occi.network.vlan	Integer	0..1	802.1Q VLAN identifier
occi.network.label	Token	0..1	Tag based VLANs
occi.network.state	Enum	0..1	Current state of the instance, possible states are active and inactive

Table 3.7: **Network Attributes** - Attributes defined for the Network type [40]

Attribute	Type	Multiplicity	Description
occi.storage.size	Float	1	Storage size in gigabytes of the instance
occi.storage.state	Enum	1	Current status of the instance, possible states are online, offline, backup, snapshot, resize and degraded

Table 3.8: **Storage Attributes** - Attributes defined for the Storage type [40]

Supporting these Resource types are the following Link sub-types :

- NetworkInterface- connects a Compute instance to a Network instance. This is complimented by the IPNetworkInterface Mixin which supports L3 and L4 capabilities. The attributes defined for the IPNetworkInterface Mixin can be seen in the table 3.9

- Storage Link- connects a Compute instance to a Storage instance. The attributes defined for the Storage Link Type can be seen in the table 3.10

Attribute	Type	Multiplicity	Description
occi.network.address	IPv4 or IPv6 Address range	0..1	Internet Protocol network address
occi.network.gateway	IPv4 or IPv6 Address	0..1	Internet Protocol network address
occi.network.allocation	Enum	0..1	Address allocation mechanism, possible values are dynamic and static

Table 3.9: **IPNetworking Attributes** - Attributes defined for the IPNetworking Mixin [40]

Attribute	Type	Multiplicity	Description
occi.storagelink.deviceid	String	1	Device identifier as defined by theOCCIservice provider
occi.storagelink.mountpoint	String	0..1	Point to where the storage is mounted in the guest OS
occi.storagelink.state	Enum	1	Current status of the instance, possible states are active and inactive

Table 3.10: **Storage Link Attributes** - Attributes defined for the Storage Link type [40]

3.3.2 pyOCNI

Python Open Cloud Networking Interface (pyOCNI) is a Python implementation of an extended OCCI with a JavaScript Object Notation (JSON) serialization and a cloud networking extension. It was developed by Institut Telecom [41]. In the pyOCNI implementation the following classes were defined:

- Availability: Represents the entity of the availability interval
- CloneNode: A networking resource of the Flash Network Slice
- CloneLink: A network link of the Flash Network Slice, see table 3.14
- FNS: A resource that provides a network service
- CloNeComputeLink: Connects a CloNeNode instance to a Compute instance
- CloNeStorageLink: Connects a CloNeNode instance to a Storage instance
- CloNeNetworkInterface: Connects a CloneNode instance to a CloNeLink instance
- FNSInterface: Connects a FlashNetworkSlice instance to a Resource instance
- Ethernet: Ethernet Mixin

- IPv4: IPv4 Mixin
- OpenFlowCloNeNode: OpenFlow Mixin that can be applied on CloNeNode
- OpenFlowCloNeLink: OpenFlow Mixin that can be applied on CloNeLink, see table 3.14
- OpenFlowCloNeNetworkInterface: OpenFlow Mixin that can be applied on CloNetworkInterface
- l3vpn_service_type: A type that is used by L3VPN Mixin to describe the service, see table 3.16
- l3vpn_service_description: A type that is used by L3VPN Mixin to identify the endpoints and the connection parameters, see table 3.13
- l3vpn_elastisity: A type that is used by L3VPN Mixin to indicate if the service supports reconfiguration and the time necessary for reconfiguration, see table 3.17
- l3vpn_scalability: A type that is used by L3VPN Mixin to indicate if the service is scalable and the time needed to deploy the changes, see table 3.18
- l3vpn: A Layer 3 VPN Mixin, it is the container of the previous mentioned types, see table 3.15

Some of these classes will not be used in the prototype, as either they refer to technologies that will not be implemented or to the fact that pyOCNI will only be used to instantiate FNSs, even though it supports management of IaaS platforms. The Kind and Mixin instances added to extend OCCI can be seen in the tables 3.11 and 3.12.

Attribute	**Value**
term	CloNeLink
sheme	http://schemas.ogf.org/occi/ocni
class	kind

Table 3.11: **OCNI Kind Definition** - [41]

Attribute	**Value**
term	L3VPN
scheme	http://schemas.ogf.org/occi/ocni
class	mixin

Table 3.12: **OCNI Mixin Definition** - [41]

Attribute	Type	Description
ocni.l3vpn.service _description.endpoint_id	String	Unique id of the endpoint
ocni.l3vpn.service _description.in_bandwidth	String	Current bandwidth for the traffic entering the endpoint
ocni.l3vpn.service _description.out_bandwidth	String	Current bandwidth for the traffic exiting the endpoint
ocni.l3vpn.service _description.max_in_bandwidth	String	Maximum bandwidth for the traffic entering the endpoint
ocni.l3vpn.service _description.max_out_bandwidth	String	Maximum bandwidth for the traffic entering the endpoint
ocni.l3vpn.service _description.latency	String	Current value for the acceptable latency
ocni.l3vpn.service _description.max_latency	String	Maximum latency that can occur in the connection

Table 3.13: **L3VPN Service Description** - Attributes defined by the L3VPN Mixin Service Description type[41]

Attributes	Type	Description
occi.core.id	String	A unique identifier of the resource inherited from Entity
ocni.core.title	String	The display name of the instance inherited from Entity
occi.core.summary	String	Short description of the resource
mixins	Mixin	Mixins associated with the resource to extend capabilities
links	Link	Connections to other resources
ocni.clonelink.state	Enum	The current state of the resource, possible states are active and inactive
ocni.clonelink.availabilty	Availability	Start and End time of the Resource
ocni.clonelink.bandwidth	String	Value of the available bandwidth in the link
ocni.clonelink.latency	String	Value for the acceptable latency in the link
ocni.clonelink.jitter	String	Value for the acceptable jitter in the link
ocni.clonelink.loss	String	Value for the acceptable loss in the link expressed in percentage
ocni.clonelink.routing _scheme	Enum	The routing scheme to be used, possible values are unicast, multicast and broadcast

Table 3.14: **CloNeLink** - Attributes defined for the CloNeLink sub-type Entity [41]

Attributes	Type	Description
ocni.l3vpn. infrastructure.request_id	Integer	Unique identifier of the request
ocni.l3vpn.customer_id	Integer	Unique identifier of the customer
ocni.l3vpn.service_type	Service_Type	Type used by the L3VPN Mixin to describe the type of Service.
ocni.l3vpn.service _description	Service _Description	Type used by the L3VPN Mixin to extend the attributes of the resource
ocni.l3vpn.elastisity	Elastisity	Type used by the L3VPN Mixin to add information about the elasticity of the resource
ocni.l3vpn.scalability	Scalability	Type used by the L3VPN Mixin similar to elastisity but regarding scalability

Table 3.15: **L3VPN Mixin** - Attributes defined by the Layer 3 VPN Mixin [41]

Attributes	Type	Description
ocni.l3vpn.service_type. type	String	The type of service, in this case L3VPN
ocni.l3vpn.service_type. layer	String	The separation layer of the VPN, in this case Layer 3
ocni.l3vpn.service_type. class_of_service	String	The service class, in this case Communication as a Service, CaaS

Table 3.16: **L3VPN Service Type** - Attributes defined by the L3VPN Mixin Service Type type [41]

Attributes	Type	Description
ocni.l3vpn.elastisity. supported	Enum	Indication if the capability is supported by the resource or not, possible values are yes or no
ocni.l3vpn.elastisity. reconfiguration_time	String	Time needed to reconfigure the resource

Table 3.17: **L3VPN Elasticity** - Attributes defined by the L3VPN Mixin Elasticity type [41]

Attributes	Type	Description
ocni.l3vpn.scalability. supported	Enum	Indication if the capability is supported by the resource or not, possible values are yes or no
ocni.l3vpn.scalabitliy. setup_time	String	Time needed for the resource to deploy the changes made

Table 3.18: **L3VPN Scalability** - Attributes defined by the L3VPN Mixin Scalability type [41]

3.4 Intralayer Communication

The DCP describes a category of protocols and interfaces that provide a mean to interact and exchange cross-administrative domain information. The DCP has an important role in the configuration of inter-domain connections between the various providers within a CloNe service. Since the various providers are operating asynchronously, the messaging system must use queues to avoid any lost messages. The message exchange pattern is based on the request-response model in which the Network Manager (NM) acts like a client and the endpoint as server.

For example, a company site located in Lisbon is trying to connect to a datacenter in Madrid, through a VPN. The DCP will be used to exchange the required information to establish a connection between the company site in Lisbon and the portuguese network provider, between the spanish network provider and the datacenter in Madrid and between the two network providers to configure the PE routers on either side, see figure 4.2.

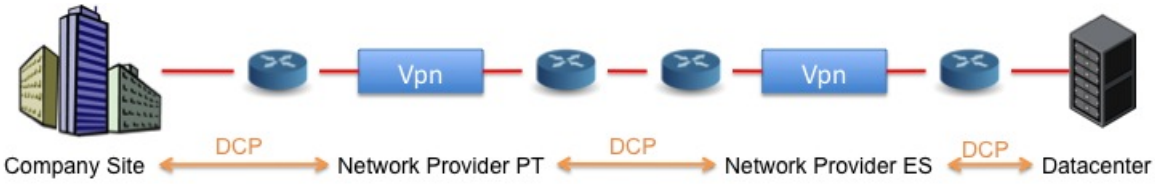


Figure 3.3: DCP Messages

3.4.1 Distributed Control Plane

The DCP protocol used in this prototype was provided by Portugal Telecom Inovação (PTIN), it's built upon the cross-platform and open-source RabbitMQ which uses the Advanced Message Queuing Protocol (AMQP) [42]. The AMQP was developed as an application layer protocol for message-oriented middleware leaving to the system designer the choice of the messaging pattern, which is called the AMQ model. The AMQ model is composed by a set of components and rules for connecting them; these can be put together in a modular fashion. There are three main types of components: the "exchange" receives messages from publisher applications and routes them to "message queues", the "message queue" stores messages until they are retrieved by a consumer application, and the "binding" defines the relation between a message queue and an exchange. The middleware server role is to accept the messages and route them to different message queues, where they are store until the consumers are ready to fetch them, the clients can be consumers, producers or both [42] (see figure 3.4).

An analogy can be made between the AMQP and an email system:

- an AMQP message is the equivalent of an email message
- a message queue is like a mailbox
- a consumer is like a mail client that fetches and deletes email
- an exchange is like a mail transfer agent
- a routing key is the equivalent of an email address

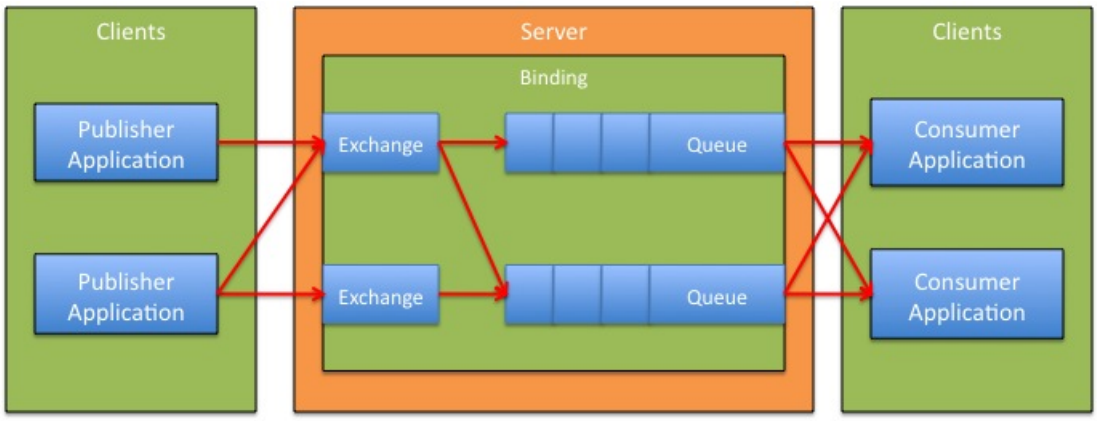


Figure 3.4: AMQ-Model

- a binding is like an entry in a mail transfer agent routing table

The DCP implementation uses a simple point-to-point routing where the routing key is the name of the message queue and the message itself is encoded in JSON. There are three defined messages in DCP: *PE Info* which is sent by the Network provider, *CE Info* which is sent by the Cloud provider, and *Config Details* which is sent by the Network provider and concludes the negotiation. In common with all the messages is the identification of the infrastructure the messages relate to, the Id of the sender and the message type Id.

PE Info

PE Info is the message sent by the Network provider to start the negotiations with the Cloud provider for the link between the CE router and the PE router. Message identifies the Network provider which is trying to establish a connection, the ISR Id which is common to the virtual network infrastructure and the cloud computing resources and the PE router which will be used to connect to the Datacenter.

CE Info

After receiving the *PE Info* message, the Cloud provider will respond with the *CE Info*, which is the message that contains the information regarding the available physical links to the specific PE router identified in the previous message. This information covers protocols supported, available bandwidth and IP addresses.

Config Details

This is the message that concludes the negotiations and it is sent by the Network provider. The message contains the options chosen from the available configurations in the previous message. The VLAN tag for the PE CE link is also included in the message.

Dependencies

There is only one library needed by DCP that is not part of a standard Python 2.7 installation, it's called *pika* which is a Python implementation of the AMQP protocol that was developed primarily for use with RabbitMQ [43].

3.5 CloNe Orchestrator

The CloNe Orchestrator represents a cross domain provider, it is responsible for interacting with the infrastructure service user and delegate the requests received to the lower providers. It is accessible through a web page where the user can submit his request, a Virtual private eXecution infrastructure Description Language (VXDL) file (see image 3.5).

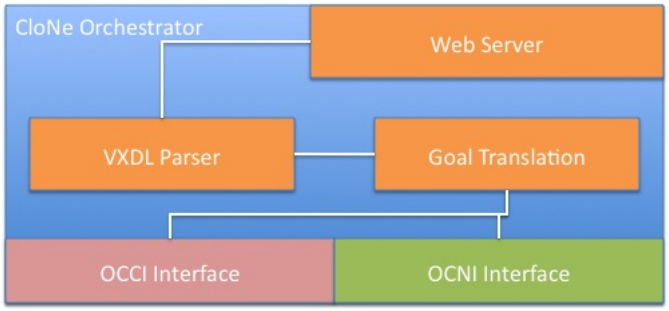


Figure 3.5: CloNe Orchestrator Modules

Goal Translation

This part of the CloNe Orchestrator is responsible for separating all the information within the ISR by administrative domains.

OCCI Interface

After the goal translation the CloNe Orchestrator will delegate the deployment of computing resources to the appropriate providers using this interface. A more detailed information on OCCI can be found in 3.3.1, while the concrete implementation will be described latter as part of the OpenNebula (section 3.6.1).

OCNI Interface

The OCNI interface is used to delegate the network resources to the network provider. This interface was described previously in section 3.3.2).

3.5.1 VXDL

VXDL [44] is a high level language to design dynamic virtual infrastructures. It was developed by Lyatiss [44] and INRIA [45]. A VXDL file contains the formal description of a

virtual infrastructure using Extensible Markup Language (XML) as syntax, the file structure is composed of four types of information:

- general description
- resources description
- network topology description
- timeline description

VXDL is still a work in progress and, in its current version it is still missing some configuration parameters to better describe all the virtual infrastructures options available. The current basic components defined in VXDL are [44]:

- vNode: virtual node, is used to describe a virtual machine
- vLink: virtual link, is used to describe a logical link
- vStorage: virtual storage, is used to describe a block storage resource
- vRouter: virtual router, is used to describe a network device, e.g.router or a switch
- vAccessPoint: virtual access point, is used to indicate the presence of an interface for communications external to the virtual infrastructure

3.5.2 Dependencies

To implement the CloNe Orchestrator web page, the library *Bottle* was used. Apart from that all other libraries are part of the standard Python 2.7 installation. Bottle is a Web Server Gateway Interface (WSGI) web framework [46].

3.6 Cloud Computing Domain

This is where all the computing resources will be allocated. Within the testbed there are two computers simulating a data center. They will be located in different endpoints of the network. The computers use OpenNebula to deploy virtual machines and a module that provides the DCP interface and configures the CE routers (see figure 3.6).

3.6.1 OpenNebula

OpenNebula was the IaaS enabling platform used in the CC domain, the version used is 3.2. It allows to instantiate configurable virtual machines using an OCCI compliant interface to receive requests from CloNe Orchestrator. To instantiate a virtual machine, the network and disk image must be deployed before (failing to provide a valid network or disk image will result in a failure to launch the virtual machine).

Initially Open Nebula only supported SQLite but since version 2.0 it supports also MySQL. The tables defined in the SQLite data base can be seen in table A.1. [26]

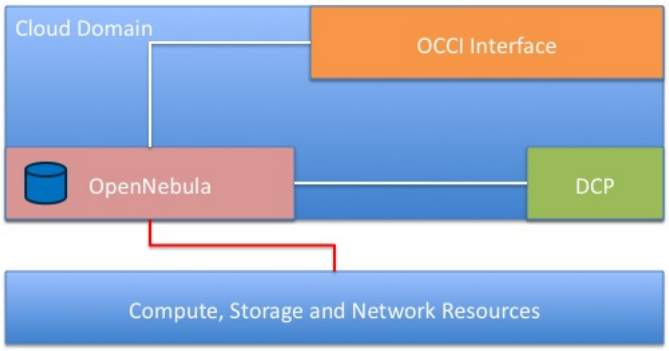


Figure 3.6: Cloud Computing Domain modules

State	Description
init	initial state when the VM is being built
pending	by default a VM starts in the pending state, waiting for a resource to run on. it will stay in this state until the scheduler decides to deploy it or the user deploys it manually
hold	the owner has held the VM and it will not be scheduled until it is released, it can be, however, deployed manually
active	the normal state during the VM life-cycle which is complemented by active sub-states (boot, running, migrate, shutdown,...)
stopped	the VM is stopped. VM state has been saved and it has been transferred back along with the disk image to the front-end
suspended	the same as stopped but the files are left in the remote machine
done	the VM has completed his life cycle, all the information regarding the VM will be kept for accounting purposes
failed	Open Nebula encountered an error during the normal execution of the VM

Table 3.19: **ON State Table** - Open Nebula possible states of a virtual machine

Table 3.19 contains the possible states of a VM in OpenNebula.

OCCI Interface

This is the interface that provides interlayer communication enabling creation, management and the erasure of the resources; it uses XML to encode the information in the requests. When creating a resource, it will respond with the Id assigned to the resource within the OpenNebula. The valid fields when building the XML file for the resources, computing (see table 3.20), network (see table 3.24) and storage (see table 3.23) can be seen below. Multiple disk and network variables within the computing type can be used, with each declaration being an additional resource will be assigned to the VM; also they have additional fields which can be seen in tables 3.22 and 3.21.

The available states to update a compute resource are:

- Stopped
- Suspended
- Resume
- Cancel
- Shutdown
- Reboot
- Done

Variable	**Description**
Id	Id of virtual machine, not valid when creating a new virtual machine
Name	Name to be assigned to the Virtual Machine
Instance_Type	Reference to the default virtual machines instances set in OpenNebula
CPU	Percentage of the Host CPU to be assigned to the virtual machine when the Instance_Type is set to Custom
Memory	Memory value in Megabytes to be assigned to the virtual machine when the Instance_Type is set to Custom
State	Using the HTTP verb UPDATE the virtual machine state will be set to this value
Disk	Represents a virtual disk assigned to the virtual machine, multiple disks can be assigned to the same virtual machine
Nic	Represents a network interface assigned to the virtual machine, multiple interfaces can be assigned to the same virtual machine
Context	It's an extension variable, within it custom variables assigned to the virtual machine will be stored in the OpenNebula database

Table 3.20: OCCI - OpenNebula Compute Type

Variable	**Description**
Storage	Identifies the disk image to be assigned to the virtual machine
Type	Label describing the type of disk images, e.g. OS, data block

Table 3.21: OCCI - OpenNebula Disk image

Variable	Description
Network	Identifies the network to be assigned to the virtual machine
Ip	Defines the Ip address of the virtual machine, it has to comply with the network address range
Mac	Defines the Mac address of the virtual machine

Table 3.22: OCCI - OpenNebula Compute type network interface

Variable	Description
Id	Id of the disk image, not valid when creating the virtual machine
Name	Name to be assigned to the disk image
Type	Label describing the type of disk images, e.g. OS, data block
Description	Short description of the disk image
Size	Size in Megabytes of the disk, it's only valid when creating a data block

Table 3.23: OCCI - OpenNebula Storage type

Variable	Description
Id	Id of the network, not valid when creating the virtual machine
Name	Name to be assigned to the network
Address	The network address, has to comply with the mask value
Size	Ip address range of the network
Public	Boolean value
Vlan	Boolean value

Table 3.24: OCCI - OpenNebula Network type

3.6.2 DCP Module

This module is deployed in the same machine that has the OpenNebula platform providing a DCP interface for intralayer communication, which will allow the connection between resources located in different administrative domains. In this case, it will be used to negotiate the CE router and PE router link, it's also responsible for the configuration of the CE router.

3.6.3 Dependencies

All the libraries used are part of the standard Python 2.7 installation except in the DCP interface that can be seen in 3.4.1

3.7 Networking Domain

This domain represents the network provider administrative domain. Is where the FNSs will be deployed. It will use the OCNI interface to receive requests from the CloNe Orchestrator, negotiate the link between the PE and CE routers using DCP and configure the VPNs that connect endpoints in the same infrastructure (see image 3.7).

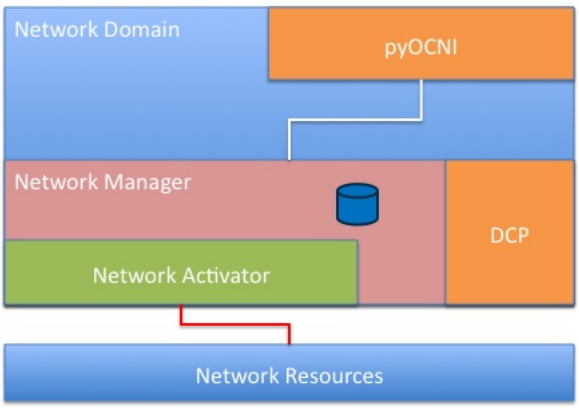


Figure 3.7: Cloud Networking Domain modules

3.7.1 pyOCNI

This module comprises an internal database, the OCNI interface and a backend for communicating with the NM module. The internal database will not be used due to the fact that the NM module already has one that is more adapted to the functionalities of the prototype.

3.7.2 Network Manager

This is the core module of the cloud networking domain and is responsible for the interconnection of all the components present in this domain. To operate on the infrastructure of the operator, XML messages will be written containing the operations and their specific

parameters. This message will be routed to the module Network Activator (NA), which acts as a web service, through its client that is installed on the same machine as the NM.

This module has a database where it is stored all the needed information regarding the network infrastructure and all the data about the deployed FNSs.

3.7.3 Considerations

Language Selection

Initially several programming languages were considered for the development of the NM; but after some discussion only two were selected to the final decision, Java and Python. To select this two various factors were used:

- ease of use
- longevity
- availability of libraries for XML creation, socket communication, webservices development,...

Eventually, the decidision factor is the fact that most of the needed libraries are part of the standard Python installation .

Although the latest version is 3.2.x, there's no backward compatibility between Python 3.x and prior versions, and due to some libraries only being available in the prior versions the NM Python version used is 2.7.

Database

To choose the relational database management system, two languages were taken in consideration, MySQL and SQLite, which are also the ones used by the OpenNebula platform. The two languages are similar due to the fact that both are based on the same programming language, Structured Query Language (SQL), both are available in many platforms and their use is widely spread (so that a lot of documentation is available online). Although either one of them could have been used, SQLite was chosen due to various factors [47]:

- Small size
- SQLite is integrated within the client application, there's no separate server process making it ideal for local storage
- The source code is in the Public Domain
- Has bindings for Python
- It's included in many platforms like Mac Os X and various Linux distributions
- The complete database can be stored in a single cross-platform file

3.7.4 Network Activator

The Portugal Telecom network is very heterogenous in terms of physical devices. For example, they have routers from multiple brands. The problem that arises is the need to know the different operating systems to configure the devices. To solve this problem it was developed a tool, NA that acts like a translator to the different devices. The NA is a mediation and service activation platform that is not capable of making any decisions, it receives configuration commands and translates them to the specific operating system of the intended device.

The NA will serve as an interface to the physical devices in the testbed for the Network Manager, allowing the latter to deploy virtual networks without the need to know specific information on how to configure different devices. Even though the testbed is only composed of Cisco branded routers, the use of NA will make the testbed a more faithful representation of the Portugal Telecom (PT) network, and also making it easier to change or add devices to testbed without having to make larger modifications to the Network Manager.

3.7.5 Dependencies

The library used in the network domain, apart from the standard Python 2.7 library is ipaddr, which is an IPv4 and IPv6 manipulation library in Python. It is necessary to have an installation of SQLite in the OS; normally it is already present in most UniX related OSs, the version used is 3.4.2 [47]. The dependencies for the DCP interface can be seen here 3.4.1.

3.8 Summary

In this chapter the prototype's architecture and all the modules present have been described as well as their intermodule relationship. Within the interlayer and intralayer communication, all the protocols and interfaces were defined by giving a thorough explanation on how the message exchange will occur and decomposing all the data that constitute the messages. All the dependencies necessary for the implementation of the modules are also presented.

Chapter 4

Implementation

4.1 Introduction

This chapter aims to describe the implementation of the prototype presented in the last chapter. Using the same organization of the previous chapter, this one will start with the CloNe Orchestrator, then the CC domain and finalizing with the Network domain. Also covered in this chapter is the Network Manager module when launched as a standalone application, which was not part of the initial requirements. An in depth review of the NM database structure and implementation is also provided.

4.2 Testbed

The testbed (figure 4.1) used to test CloNe is a small representation of a common network provider infrastructure. At the core of the network there are four routers, Earth, Saturn, Jupiter and Titan, using a BGP with MPLS configuration. The core routers are connected to four PE routers in a one to one relationship, Venus, Mars, Uranus and Neptune, in which only Uranus and Mars are capable of deploying Layer 2 VPNs, while all of them support Layer 3 VPNs. Eris, Rhea, Dione and Pluto represent the CE routers and all of them are directly connected to the PE routers in a one to one relationship. The node Tom is where the CloNe broker, pyOCNI and NM will be deployed. The node NA hosts the Network Activator virtual machine. The nodes Moon and Leo represent the data centers running each an instance of OpenNebula version 3.2.

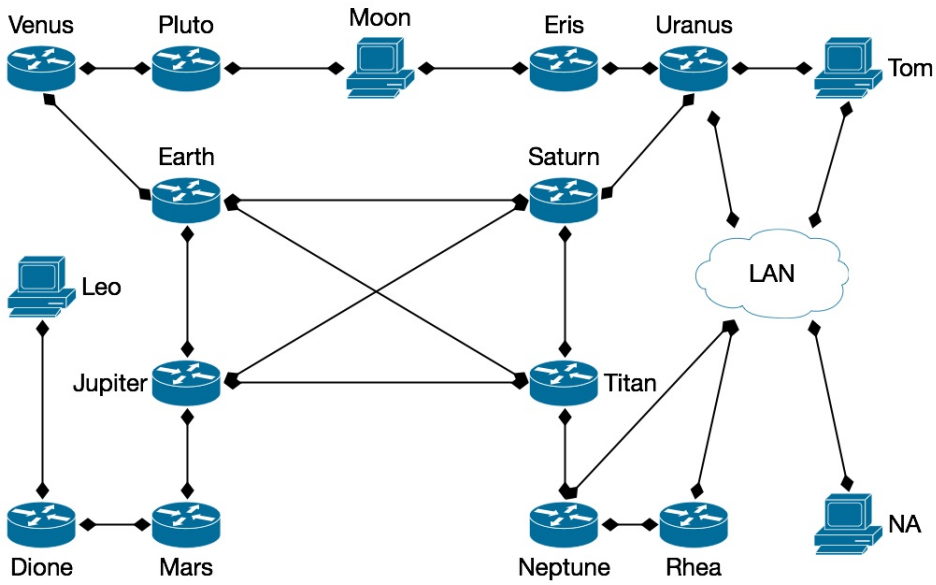


Figure 4.1: CloNe Testbed

4.3 CloNe Orchestrator

The main function of the CloNe Orchestrator is to receive the infrastructure service user request and disassemble it for delegation to the appropriate providers. The user can access the CloNe Orchestrator through a webpage where he can submit requests. A request is a XML file written in VXDL.

4.3.1 Web Page

The web page of the CloNe Orchestrator is built on top of the *Bottle* framework [46], it uses decorators to bind pieces of code to URL paths [46]. In Python, functions are objects and they can be passed as arguments just like a string or an integer. A decorator in Python is a function that takes a function object as an argument and returns a function object as a return value [48].

The *run* function in *Bottle* starts a server instance and it is where the base address for the web page is defined. On top of the server instance, a new route was defined, "upload", and it is where the CloNe web page can be accessed; its complete URL is *http://base_address:port/upload/*. Bounded to this URL are two decorators that implement the HTTP request methods GET and POST: GET draws the Hyper Text Markup Language (HTML) page in which the user can submit his request; and POST processes the submit action.

4.3.2 VXDL Parsing and Goal Translation

To parse the VXDL [44] files, which are written in XML, the module ElementTree [49] is used; ElementTree is part of the Python's standard library since version 2.5. Each XML file is loaded as an Element type, which is a container designed to store hierarchical data structure. Each Element type variable has a number of properties associated with it [49]:

- Tag - string identifying the Element
- Attributes - each Element can have one or more attributes, which consist of a string key and a corresponding value
- Text - a text attribute, which can be used to hold additional data
- Child Elements - other Elements appended to the parent element

Using iterators, standard elements defined in VXDL, such as vRouters or vNodes, are retrieved from the file. Each single element attributes will then be processed and stored. An iterator is an object which allows a programmer to traverse through all the elements of a collection.

The parsing of an VXDL request starts by retrieving all the endpoints associated with the FNS. Each of the endpoints constitutes a domain, which is a partial request of the entire infrastructure. The only entities that can be connected to an FNS are vAccessPoints and vRouters: in the first case it can be, for example, a connection to a costumer site; in the second case, it means it will be a connection to a data center. In case of a connection to a virtual router, it will then retrieve all the virtual links directly connected. Links can only point to a virtual node representing a virtual machine. A vAccessPoint can also be connected directly to a virtual machine, which means that virtual machine has public access.

Goal Translation

The goal translation starts by dividing the elements in the file by domain, where a domain is a segment of the CloNe service that can be physically separated from other segments. A domain can be:

- A virtual machine or an array of virtual machines
- A VPN
- An access point directly connected to a VPN

With all elements grouped into domains, see figure 4.2, the following step is to retrieve the attributes that can limit the choices of placement of the resource. These latter can be divided into three categories:

- geographical- within the VXDL file location preferences can be set for the various elements. When the goal translation is taking place, it will try to match the location attribute with a location set for the data centers.
- technical limitations- the current virtual networks are based on layer 3 VPNs without QoS enforcement which together with the fact that most endpoints are only connected by one physical link, make impractical the use of factors like bandwidth, latency or jitter in the goal translation.
- physical resource efficiency: although the NM and OpenNebula have means to monitor the usage of physical resources currently it is not defined in the architecture the mean to exchange this information with CloNe Orchestrator.

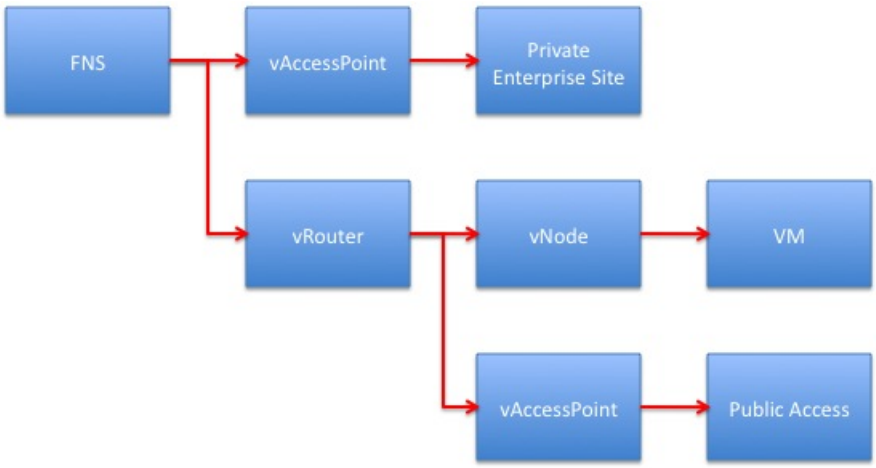


Figure 4.2: Domain identification

4.3.3 OCCI Interface

The OCCI Interface [18] is used to communicate with the OpenNebula [26] (detailed information regarding the OpenNebula implementation of OCCI is provided in 3.6.1). The module ElementTree is also used here to write the XML message, the message itself is written on top of the root Element, then the root Element will be wrapped in an ElementTree instance. The ElementTree class represents an entire element hierarchy and provides support for serialization to XML; using the write method on this class writes the ElementTree to a file. After the file has been written, it can be sent to an OpenNebula instance using cURL [50]; cURL is a command line tool for transferring data using various protocols, in this case HTTP.

4.3.4 OCNI Interface

The OCNI Interface [41] is used to communicate with the Network Operator (detailed information regarding the pyOCNI is provided in 3.3.2). The module *json* which is part of Python standard library, was used. This module uses dictionaries to write the message while maintaining hierarchy consistency; afterwards the method "dumps" serializes the root dictionary as a JSON formatted string that can be written into a file. After being written, cURL is used to send the data to OCNI server located on a Network domain.

Dictionaries is the Python's name for associative arrays, they are similar to lists, both provide indexed access to any type of object, except while lists uses integers as indices to values. Moreover, dictionaries can accept any type of object as an index. Since a dictionary can contain another dictionary, the mean to construct a hierarchy within the message is provided.

4.4 Cloud Domain

The Cloud domain consists on a DCP module and an OpenNebula instance. The OCCI interface in OpenNebula receives the requests from the CloNe provider, while the DCP module

negotiates the link to the FNS and configures the CE routers.

4.4.1 DCP

The negotiations start with the arrival of the *PE Info* message; afterwards the available links and configuration options are written in a JSON file, which is the *CE Info* message. The data sent in *CE Info* is based on the provider which sent the previous message. Using the AMQP protocol, the JSON file which composes the *CE Info* message is forwarded to the provider, which in its turn will respond by sending the *Config Details* message concluding the negotiations. To be able to configure the CE router, besides the information provided in the *Config Details* message, the VLAN Id of the network interface must be fetched from the OpenNebula database. The CE routers configuration will be done by establishing a telnet session with the routers.

The DCP module within the Cloud Domain can be divided in two parts, the communication between providers and the integration with the OpenNebula platform.

Communications

The library *pika* [43] provides an API to implement the AMQP [42], in this case RabbitMQ [51]. First a connection must be established with the RabbitMQ server. The function *ConnectionParameters* returns an object with all the options which are involved in creating the connection. All the options are set to default except the host which contains the IP address of the machine where the RabbitMQ server is launched. This object created by the function *ConnectionParameters* is passed to the function *BlockingConnection*, which will establish the connection with the RabbitMQ server. Although there are other options to connect with the server for more complex implementations this one was used for its simpler approach. After a successful connection has been established, a channel must be opened before declaring a queue: the function *queue_declare* receives the consumer tag for the queue as input and, if needed, it will create the queue. To start consuming messages from the queue, the *callback* must first be bounded to the messages for the consumer tag, then a request to start a consumer will be sent to the server, and this consumer will be available as long as the channel is open.

To send messages to other providers another connection is established with the RabbitMQ server using the same parameters as before. After establishing the connection, a channel will be created so that a message can be sent using the method *basicPublish*. The message will be routed by the broker to the consumers associated with the queue [43].

OpenNebula Integration

OpenNebula allows the customization of OCCI server using the templates files, which are written in Ruby [52]. The *common.erb* file was changed so that the ISR Id present in the OCCI request can be stored in the database.

The OpenNebula database will be accessed using pysqlite2 API [53]. Using the method *execute*, it will allow to interact with database using SQL commands, which is used to fetch the Network_pool table present in OpenNebula database. Afterwards each row in the table is processed individually until there is a match for the ISR Id. First, the content which is in XML format is parsed using the ElementTree library; second, using the method *find* the

infrastructure Id and the VLAN Id are retrieved. When the correspondent infrastructure id is found, the VLAN Id will be stored and the database connection is closed.

4.5 Network Domain

The Network Domain is composed of four main parts: the pyOCNI module [41] which implements the OCNI interface; the DCP module, which allows inter-domain communications; the NA, which configures the devices in the network infrastructure; and the NM, which integrates all the before mentioned modules.

4.5.1 OCNI Interface

The OCNI interface is provided by the pyOCNI module which was treated in this work as a black box. pyOCNI provides a customizable backend which allowed the integration of the Network Manager. Initially the communication between pyOCNI and the Network Manager was done using sockets, but since both will be deployed in the same machine, there is no need to consume extra resources. Currently, the Network Manager works as a library which is called by pyOCNI. The backend for Layer 3 VPNs represents a class of objects with four methods defined in direct relationship with the HTTP methods: "create", "read", "update" and "delete".

Whenever an OCNI request arrives at pyOCNI, the later will parse it and provide access to its content in the backend, which will then send it to the function *new_ocni_request* defined in the Network Manager. Also implemented in the backend of the module pyOCNI is the deletion of a previous created infrastructure; the procedure is equal to the one used in the case of a new infrastructure except that in this case only the ISR Id and the VPN routing and forwarding table (VRF) Id will be sent to the *delete_infrastructure* subroutine.

4.5.2 Network Manager

When the subroutine *new_ocni_request* is called by the pyOCNI module, the NM will start by checking in the database for the existence of another entry with the same ISR Id to avoid duplicated entries. Afterwards it processes all the endpoints and retrieves their respective DCP addresses from the database. In case there is no DCP address available, the NM will immediately send the request to the NA to deploy a VPN in that endpoint using default values, while saving all the configurations in the database. If there is a DCP address available, a *PE Info* message is sent to the respective endpoints signaling the beginning of PE-CE link negotiations. All the other parameters received in this function will be stored in the database for later use (see figure 4.3).

Figure 4.3: Subroutine new_ocni_request

The selection of the PE router is made by evaluating two variables, the layer type and the interface type, within the available links to that endpoint. Since there are only two routers

in the testbed that support layer 2 VPNs, the NM will give precedence to the ones that only support layer 3 and the serial interfaces will be discarded because they do not support VLAN tags. It should be mentioned that the bandwidth does not constitute a selection criteria, because it is not being enforced on the logical links. The PE selection starts by retrieving the endpoint reference using the endpoint unique Id to be able to get a list of available hard links from the NM database. Then, using the variables mentioned before, a physical link will be chosen (see figure 4.4).

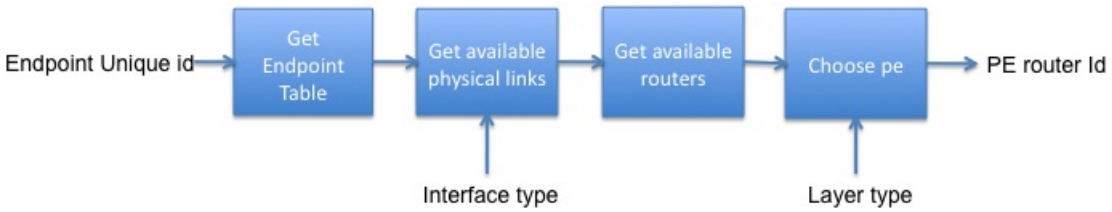


Figure 4.4: PE Selection

Upon the arrival of the *CE Info* message, the *callback* function will be called. This function will start by parsing the message and retrieve all the information, and will then calculate the various parameters to configure the routers:

- route distinguisher - the route distinguisher must be unique within the Network provider infrastructure; the number used will be the result of the sum between 1500 and the number of VPNs instantiated. To preserve the integrity of the network, values Below 1500 are reserved for manual configurations
- route target - the route target will use the same method as the route distinguisher
- VLAN tag - the VLAN tag used will be the result of the sum between 1500 and the total number of VPNs deployed in that particular PE. To preserve the integrity of the network, values Below 1500 are reserved for manual configurations.

After all the necessary parameters have been gathered, the NM will send a request to the NA so that the endpoint can be added to the VPN. All the endpoints that are connected to the VPN are processed individually and, within a request ,there can be endpoints that have a DCP module and others that do not. The difference is that, since there is no negotiation through DCP, these endpoints will be configured with default values for IP addresses and protocols, while the remaining parameters are calculated in the same manner as described before. Afterwards, the NM will store all the parameters used in the database and it will also update the information stored in the database regarding the physical infrastructure. When the VPN has been deployed and the information stored in the database, the NM will send the last DCP message, *Config Details*, informing the endpoint of all the parameters used to configure the VPN.

Due to the fact that the NM is multi-threaded, it could be possible to have multiple functions accessing the database concurrently, which is not supported by SQLite [54]. To avoid this scenario, every time the database is opened, a lock is enabled prohibiting other functions to access it. These functions will remain on hold until the lock is released. The lock mechanism is enabled by the *Threading* library, which is part of Python standard library.

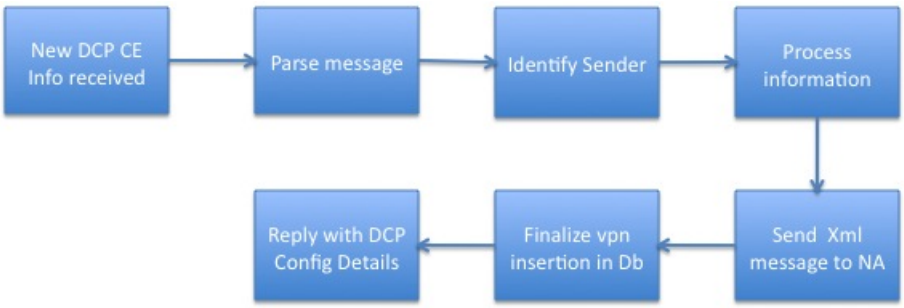


Figure 4.5: Callback

First, a lock object must be created with two associated methods, *acquire*, which enables it, and *release*, which as the name suggests releases the lock.

Delete Infrastructure

When the pyOCNI calls the routine *delete_infrastructure* it sends the variables ISR Id and the Id of the VPN. The database will provide all the information necessary for the deletion of the logical links associated with the VPN; then the NM can send the request to the NA so that the VPN can be removed. Afterwards, the NM will update the information regarding the physical links and delete all entries from the database associated with this infrastructure (see figure 4.6).

Figure 4.6: Delete Infrastructure

DCP

The procedures to connect to the RabbitMQ server are equal to the ones used in the Cloud Domain, see 4.4.1. Likewise the messages will be written using the *json* library. The callback function bounded to the consumer tag in the DCP communications is launched in another thread. This function is launched whenever a *CE Info* message is received from the Cloud provider.

4.5.3 Network Activator

The NA tool works as a webservice using the protocol Simple Object Access Protocol (SOAP) [55], and to communicate with it there are two available options:

- direct communication using SOAP
- use the NA client to wrap a XML message inside the SOAP message and send it to NA

Although direct communication could have been used, it would still imply to write first a XML message and wrap it in a SOAP message. To avoid unnecessary work and find a library capable of importing objects directly into a SOAP message, the second option was chosen. The NA client has to be installed in the same machine of the NM with an available route to the machine where the NA is deployed, in case of being in different machines. To use the NA client, two files must be created: one is the XML request message, and the second one is the properties file with the address of the NA webservice. The NA client can be called through terminal with the XML request file as an argument, and the properties file must be place inside the same directory of the request. After the message has been sent, NA client sends to the standard output the telnet sessions made by the NA to the intended routers. It will also be printed a XML message regarding the success or failure of the communications between NA and routers. The NM will store all the output in standard text files thus providing a mean to parse the information sent by the NA.

The NM can make four kinds of requests to NA: the creation of a layer 3 VPN with either Open Shortest Path First (OSPF) or Routing Information Protocol (RIP) protocol; a layer 2 VPN; the deletion of layer 2 VPNs; and the deletion of layer 3 VPNs.

XML Request Files

The structure of the XML files is very similar to all the requests. It covers all the common and specific information regarding the requests. The message root element is the *service-descriptor*. It contains the field *rfsMplsIpVpn* field that identifies the operation: it can be "createVpn", "createVpnL2", "removeVpn" or "removeVpnL2", while the *genericIpRouter* field stores the information regarding the configuration of individual routers. Inside the same message, it can send multiple routers configuration (see figure 4.7). Within the field *genericIpRouter*, the attribute *lrVersion* indicates the message structure version regarding the specific operation, the currently used versions are:

- Operation "createVpn" - v1.1 for the RIP protocol and v1.2 for the OSPF protocol
- Operation "createVpnL2" - v1.0
- Operation "deleteVpnL2" - v1.0
- Operation "deleteVpn" - v1.2

The *managementInfo* indicates the context of the request, while the info field is where the individual router is identified and where the specific request information is located. To identify the router, three parameters are used: the device type, in this context is always Router; the device name, which is unique to all the routers; and the IP address that can be used to connect to the device. The *otherInfo* field is where the specific information to the request is presented; it has different parameters depending on the operation and the lrVersion. Although this field presents different parameters depending on the situation, there is one field within it that is common to all: it is the *logicalinterface*, which is where the virtual interface is identified and defined. The operations defined are:

- Operation "createVpn" - identifies the logical interface to be created and its IP address, the network address and its mask, and also the VLAN tag to be used

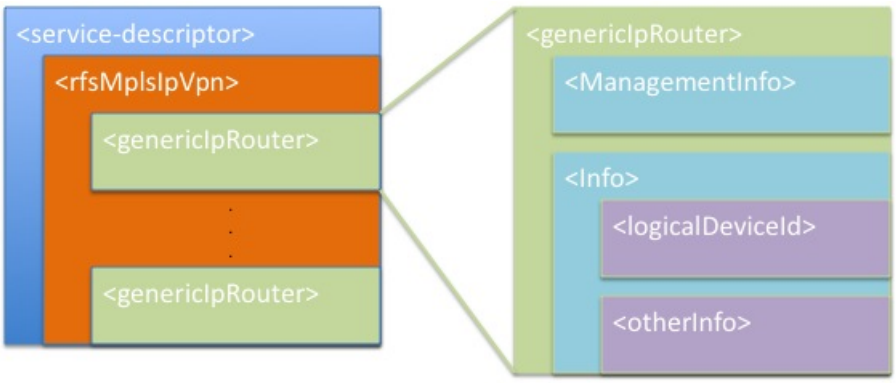


Figure 4.7: XML structure

- Operation "createVpnL2" - identifies the logical interface to be created plus the VLAN tag to be used
- Operation "deleteVpnL2" - identifies the logical interface to be deleted(it is a combination of the physical interface and the VLAN id)
- Operation "deleteVpn" - identifies the logical interface to be deleted

In a request to create a layer 2 VPN, the *otherInfo* will also contain the *vcId*, which is the common identifier that is used by each end of the virtual circuit to identify it, and the *peerIP*, which is the IP address of the router located in the other endpoint [56]. In the case of being a layer 3 VPN instead of *vcId*, it will be used the *vrfId* plus the route distinguisher and route target: a VRF is a routing table instance that only exists on PEs; the route distinguisher is a number which identifies the VPN in a provider's network; and the route target indicates the VPN membership and allows VPN routes to be imported and exported into or out of VRFs [57]. If the protocol used in the layer 3 VPN is OSPF, then an additional field will be provided indicating its *Id*; if the protocol is RIP nothing else will be added.

The information for the deletion of VPNs is very similar for both cases, layer 2 or layer 3. Besides the logical interface explained before there is an additional field identifying the VRF or the virtual circuit, depending on whether it is a layer 3 or layer 2 VPN.

4.5.4 Network Manager Database

The NM database was created using the relational model: each element in the table is referenced by its own unique Id and each table represents only one thing, e.g. a virtual infrastructure, a link. The advantage of using this model is that the perceived structure of the database can change without needing to alter the database schema. For example, currently the infrastructure and VPN tables have a one-to-one relationship, but in the future this relationship can change to a one-to-many relationship [58].

Structure

The database has two functions, the first one is to store a complete description of the network infrastructure while keeping updated information regarding the usage of physical

resources; the second one is to store all the necessary information regarding the FNSs. In the figure 4.8, it is pictured the complete structure of the database: the tables in blue are related to the physical resources, while the green ones are related to the virtual resources.

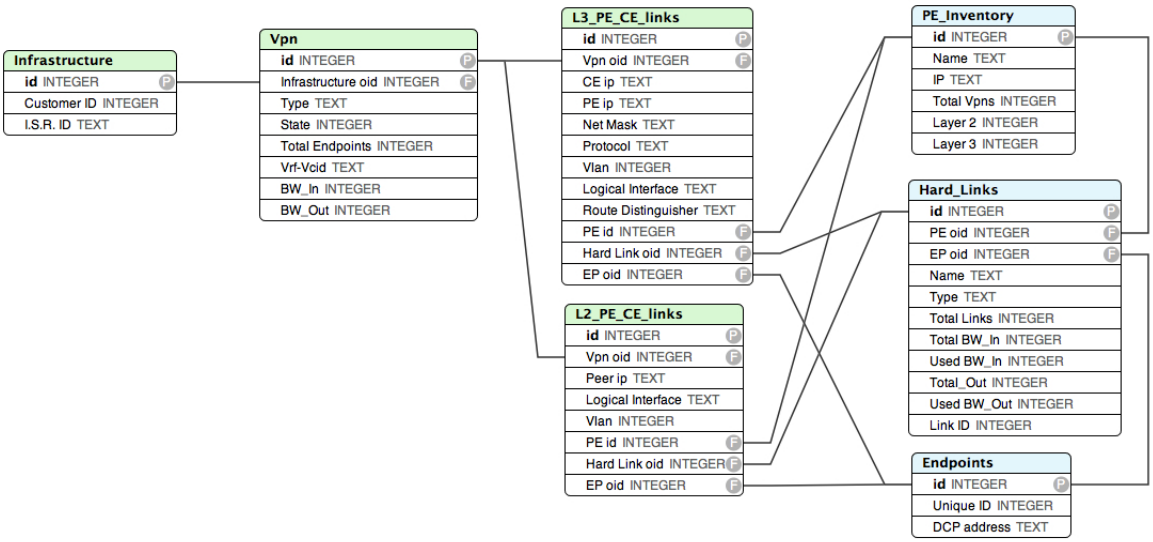


Figure 4.8: Database structure

Physical Network

In this section it is presented the various information stored regarding the physical resources in the operators network. The PE Inventory (table 4.1) has a complete list of all the routers located on the edge of the providers network. The core routers were not added to this table because, although they are part of the FNS there is no control on the path between PEs. The Endpoints table (table 4.3) stores a complete list of all the endpoints present in the network, while the Hard Links table (table 4.2) has a complete list of all the physical links that connect an endpoint to a PE. Although the Endpoints table is present in this section (it is associated with the physical network and it is important to provide a complete description of the providers network) its purpose is somewhat different from the other tables. The other tables, besides describing the infrastructure, they also keep updated information on the physical resource usage, while the Endpoints table function is similar to a personal organizer.

Virtual Network

The purpose of the following tables is to store all the information used to create FNSs and also provide the means for their removal. Currently an infrastructure can only be associated with a single VPN but because in the future that may change, the Infrastructure table was separated from the VPN table (tables 4.4 and 4.5). These two tables combined store the general attributes of the FNS, and although they are separate tables when the NM interacts with the database, accessing one of these tables normally implies accessing the other one.

Variable	Info
Id	reference for the PE router, primary key
Name	name of the router, necessary for the NA request message
Ip	ip address of the router
Total Vpns	number of vpns instantiated in this router
Layer 2/3	information on the router capability to deploy layer 3 and/or layer 2 vpns (possible values: 0 or 1)

Table 4.1: **PE Inventory** - Stores the information about the providers edge routers

Variable	Info
Id	reference for the physical link, primary key
PE oid	reference for the PE router its connected to
EP oid	reference for the endpoint its connected to
Name	name reference of the interface (ex: F0/1 Fast Ethernet 0/1)
Type	type of the physical interface, Serial or Ethernet
Total Links	the number of logical links on top of the physical link
Total BW In/Out	total bandwidth capability of the physical link
Used BW In/Out	total bandwidth allocated by the logical links

Table 4.2: **Hard Links** - Stores the information about the physical links available in the PE routers

Variable	Info
Id	reference for the endpoint, primary key
Unique ID	the Id for which is known by CloNe provider
DCP Address	the address necessary for the DCP negotiations

Table 4.3: **Endpoints** - Stores information about the several endpoints the CloNe provider can ask a connection to

The Layer 2/3 PE-CE links table (tables 4.6 and 4.7) serve two purposes: one is to store the specific parameters used to configure the PE CE links, while the second one is to provide a connection between the database tables regarding the physical network and the tables that address the virtual networks.

Variable	Info
Id	reference for the infrastructure, primary key
Customer ID	numeric id of the client
ISR ID	Infrastructure Service Request ID

Table 4.4: **Infrastructure**- Stores basic information regarding the virtual infrastructure

Implementation

The database is created using the method *connect* which returns an object that represents the database. The method receives as input the file name where the database will be stored;

Variable	Info
Id	referance for the vpn, primary key
Infrastructure oid	reference to the infrastructure id
Type	layer type of vpn, layer 2 or layer 3
State	current status of vpn, for now theres only two states, up or down
Total endpoints	number of endpoints covered by the vpn
Vrf-VcId	vrf id of the vpn layer 3 or vcid of the vpn layer 2
BW In/Out	used bandwidth by the vpn

Table 4.5: **Vpn**- Common information about vpns whether they are Layer 3 or Layer 2

Variable	Info
Id	reference of the layer 3 link, primary key
Vpn oid	reference to the vpn the link is associated to
CE Ip	ip address of the interface of the CE router in the CE-PE link
PE Ip	ip address of the interface of the PE router in the CE-PE link
Netmask	network mask for the network in the CE-PE link
Protocol	protocol used in the CE-PE link
Vlan	vlan id of the interface of the PE router
Physical Interface	physical interface of the PE router used in the CE-PE link
Route Distinguisher	route distinguisher and route target associated with vpn
PE oid	reference to the PE router where the logical link was instantiated
Hard Link oid	reference to the physical link where the logical link is located
EP oid	reference to the endpoint the vpn connects to

Table 4.6: **L3 PE CE Links**

Variable	Info
Id	reference of the layer 2 link, primary key
Vpn oid	reference to the vpn the link is associated to
Peer Ip	ip address of the other PE router used in this layer 2 vpn
Physical Interface	physical interface of the PE router used in the CE-PE link
Vlan	vlan id of the interface of the PE router
PE oid	reference to the PE router where the logical link was instantiated
Hard Link oid	reference to the physical link where the logical link is located
EP oid	reference to the endpoint the vpn connects to

Table 4.7: **L2 PE CE Links**

if the file does not exist this method will create it. Once the connection has been established, a cursor object is created. Associated with this object there are two methods that will be used to perform SQL commands, *execute* and *executemany*: the first one executes a single command while the second one executes the command multiple times, e.g. when storing an

array into the database. Any change made to the database will only be stored when the method *commit*, which belongs to the connection object, is called. To close the connection to the database one must use the method *close*.

Every time the pyOCNI is launched, a verification will be performed on the existence of the database. If there is no database file present, a script will run in which all the tables mentioned before will be created. Besides creating the tables within the database, the tables that keep updated information on the physical resources will also be filled with the default values used to describe the network infrastructure.

4.6 Standalone NM

The NM primary objective is to deploy OCNI requests, but this might not always be the case, e.g. when testing other modules or trying to detect a malfunctioning network device. Although this might seem trivial some extended protection regarding the integrity of the database should be enforced avoiding possible traps like having duplicated ISR Ids.

Menu Options

The menu will only appear when the network manager is launched as a standalone application. The menu options available are:

- create vpn layer 2
- create vpn layer 3
- delete vpn
- exit

4.6.1 VPN Deployment

The *ISR Id* will be the first parameter to be asked whether it is layer 2 or layer 3 VPN, which will then be checked for duplicated values. In case of the layer 3 VPNs the number of endpoints will be asked, while in layer 2 this value is locked to two endpoints. If it is a layer 3 VPN, to each one of the endpoints, the IP address of the logical interface will be asked as well as the protocol, OSPF or RIP, which will be used in the PE-CE network. The route distinguisher/target and VLAN for stability purposes will be locked to sub 1500 values, upper values are reserved for OCNI requests. Since this is a manual configuration, the database will be checked to see if these values are already in use. If it is a layer 2 VPN, the only options available will be the virtual circuit Id and the VLAN Id, which will also be checked in the database for duplicate values. The choice of endpoints is locked, because within the testbed, there are only two available endpoints for layer 2 VPNs.

The rest of the procedure to deploy the VPNs will be similar to the ones created through OCNIs requests.

4.6.2 VPN Removal

The process of manual deleting a VPN is similar to deleting a VPN through the OCNI: the *ISR Id* and *VRF* or *vcId* will be asked to identify the infrastructure to be deleted. The

only difference is that the *ISR Id* will be checked for existence, and in the case of using OCNI, that evaluation is performed by the pyOCNI module.

4.7 Summary

In this chapter the implementation of the three domains present in the prototype was described, covering their sequence of actions, structure and methods and classes used. The NM database is deconstructed in this chapter to provide an in-depth knowledge of its structure, along with its implementation.

Chapter 5

Tests &Analysis

5.1 Introduction

In this chapter the CloNe prototype will be analyzed through performance tests to obtain a better view on all the processes involved in a deployment of a CloNe service.

The NVSS platform will be tested regarding different types of traffic. Before this platform can be used in the prototype, an evaluation of its performance has to be made to assess the need for further optimization. The TCP tests will evaluate the bandwidth when varying the number of Virtual Routers (VRs), and its degradation when faced with higher CPU loads. UDP tests are also performed to evaluate the influnece of virtualization on the delay and jitter, assessing if the platform is ready to deploy network services with jitter requirements.

This chapter will start with the analysis of the CloNe prototype and it will finish with the tests performed on the NVSS platform.

5.2 CloNe Analysis

This series of tests are meant for a qualitative analysis of the prototype. These tests intend to show a temporal analysis of all the processes involved in the instantiation of a CloNe service, while trying to identify the presence of possible bottlenecks. When necessary it will be identified the situations in which the tests may be adversely affected by limitations of the physical devices in order to separate the analysis of the prototype from the underlying infrastructure.

5.2.1 Testbed

The testbed used in this section is the one described in 4.2. The specification of the equipment present in the testbed can be seen in table 5.1.

5.2.2 Methodology & Results

The evaluation of the proposed solution was divided in two parts: the first part consisted in the instantiation of a network service, while the second part consisted in the deployment of a VM. In the setup for the first test, a VM was deployed in the node Moon, while in the node Leo an interface was configured to use VLAN tags. Instead of deploying a VM in Leo, a virtual interface was created to access the VPN just like it is done to the VM in Moon. A

Node	Moon	Leo	Tom
CPU Model	Intel Core 2 Duo E6750	Intel Core 2 Duo E6750	Intel Pentium 4
CPU Freq	2.66GHz	2.66GHz	3.2GHz
CPU Cores	2	2	2
CPU Threads	2	2	2
HDD Memory	141GB	142GB	71GB
RAM Amount	3GB	2GB	2GB

Table 5.1: **CloNe Machine Specification**

script in node Tom calls the NM method *new_ocni_request* to configure a VPN between the VM in node Moon and the node Leo. Using the *time* library, the duration of all the steps necessary to process the request were recorded. Immediately after processing the request, another script in Leo pings the interface of the VM until there is connectivity between the two nodes. These two scripts ran 100 consecutive times. In the table 5.2, the three main operations involved in the request are showed. The database access time comprises all the accesses made to the database, while the DCP communications show the time spent building the messages *PE Info* and *Config Details* and sending them to Moon and Leo. The NA communication shows the time spent building the two XML messages, one for each endpoint, and the duration of the communication with the NA.

Operation	Duration (s)	Confidence Interval (90%)
Database Access	29.316×10^{-3}	0.068×10^{-3}
DCP Communications	15.767×10^{-3}	0.036×10^{-3}
NA Communication	18.713	1.334×10^{-3}

Table 5.2: Network Manager Analysis

The NA communication is the most time consuming process with the other two processes having residual durations during the request.

In the second part of the analysis, only 10 deployments of the VM were used, because the deployment of the VM is mostly affected by copying the disk image. The VM is based on Ubuntu OS version 12.04 LTS using 10GB of disk space.

In the table 5.3 the Network Manager field shows the time difference between receiving the request and the last Config Details message, which marks the conclusion of the request. The Connectivity value shows the time between finishing the request by the NM and the time of the first PING reply. This value shows the time necessary for the routers to exchange the routes within the VPN. The OpenNebula field shows the time necessary for the VM to be booted after the command to deploy the VM.

The launch of the VM is the most time consuming process within a CloNe request, mostly because of the time needed to copy the disk image. The VM deployment time is largely affected by the time needed to copy the disk image. The time necessary to deploy other OSs is largely dependent on the disk image size, which would mostly reflect the hardware capabilities. The time needed to exchange routes is mostly affected by the routers capabilities and, although the testbed itself is quite small, newer routers in a network provider infrastructure are capable of smaller times.

Phase	Duration	Error Interval (90%)
Network Manager	18.779	1.346×10^{-3}
Connectivity	62.23	0.378
Total	81.009	-
OpenNebula	378.4	1.734

Table 5.3: CloNe Analysis

5.3 CloNe Tests

The aim of this test is to evaluate the response of the Network Manager when receiving simultaneous requests. The NM has a lock mechanism that prevents simultaneous accesses to the database: when a request is being serviced, most of the steps involve operations with the database. This test will be able to tell if the lock mechanism hinders performance, and also the overall behavior of the NM when under stress.

5.3.1 Methodology & Results

The tests consist in sending simultaneously a varying number of requests to the NM bypassing pyOCNI. The number of requests ranges from 2 to 5, in each value the test is run 10 times. Upon receiving the requests, the NM will perform exactly like it was receiving them from pyOCNI, meaning that it will use the DCP to negotiate the links in the PE-CE connection.

In the Network domain two factors are tested:

- Delay Time- the time it takes the NM to start processing the request after it has been sent. Ideally this should be the same as the local time, but due to the lock mechanism, the request may be put on hold until the lock is released.

- Total Time- the time it takes the NM to complete the request after it has been sent. This value is affected by the delay time and the lock mechanism. In this case, after sending the *PE Info* message to the datacenter, it will receive the *CE Info* message, but if the NM is busy processing another request, the message will not be consumed.

When two requests were sent simultaneously, most of the times the second request was put on hold. The cause of the latter is that a *CE Info* message was received before the NM could start process it (figure 5.1). The time it takes to complete the first request is very close to the time measured in the previous section for a single request.

In the case with three requests, the results are very similar to the previous one (figure 5.2). With the first request being processed within a timescale of a single request and leaving the remainders "fighting" for the access to the NM. With four requests sent simultaneously the second, third and fourth request are processed consecutively; only between the first and the second request there is a significant delay time (figure 5.3). In the following case, in contrary with the previous ones, the delay registered for the second request is reduced significantly (figure 5.4).

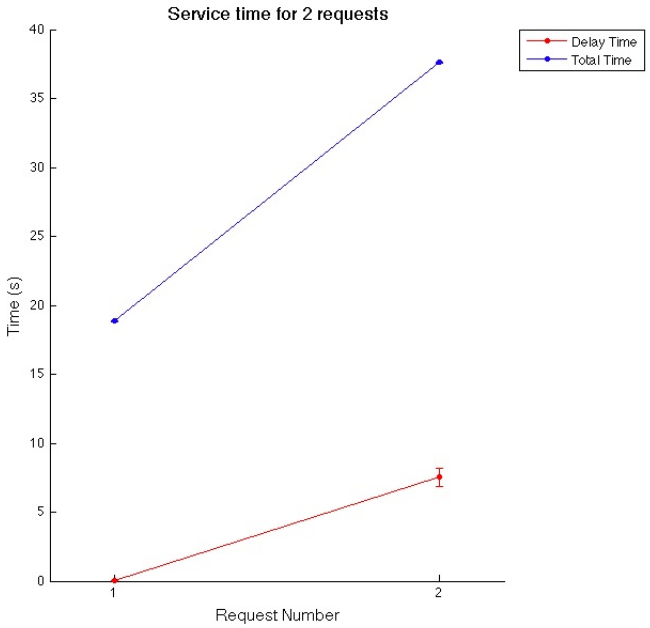


Figure 5.1: Delay and Total service time for 2 simultaneous requests

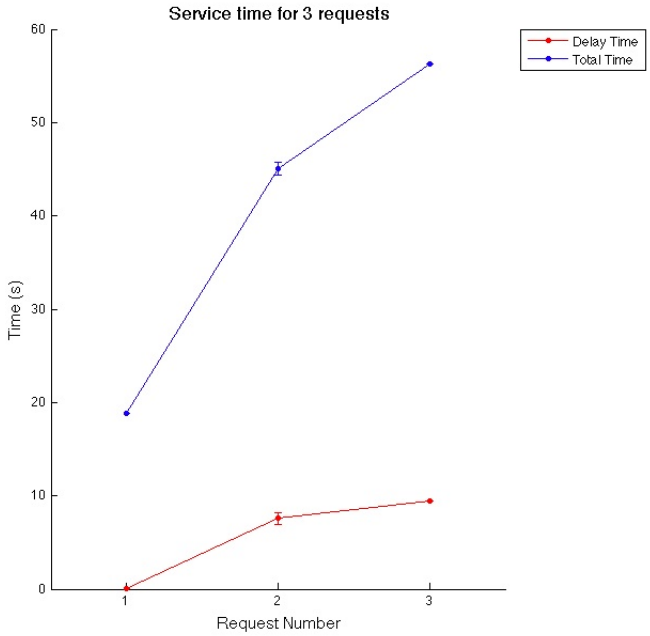


Figure 5.2: Delay and Total service time for 3 simultaneous requests

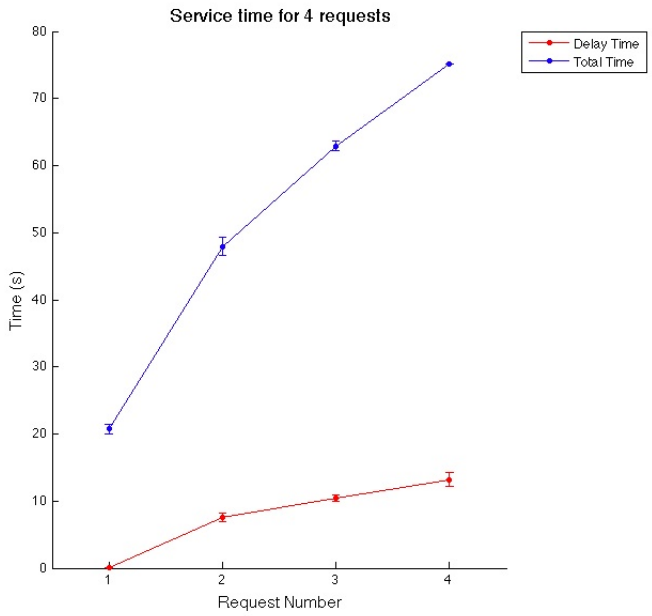


Figure 5.3: Delay and Total service time for 4 simultaneous requests

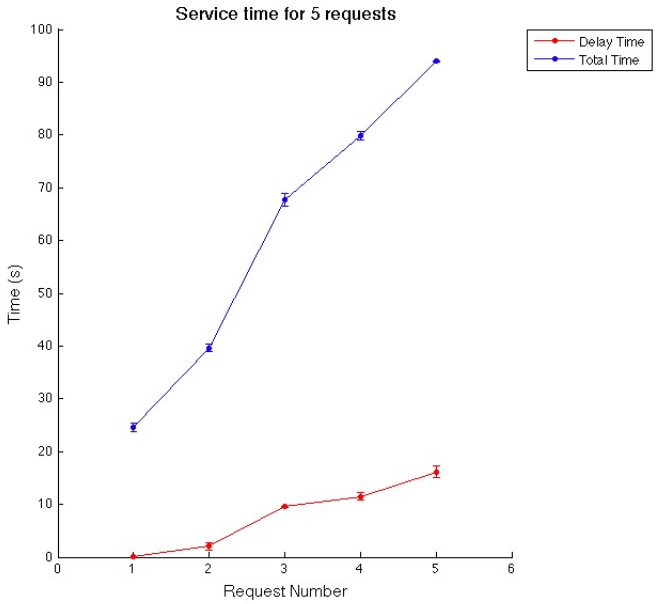


Figure 5.4: Delay and Total service time for 5 simultaneous requests

In the datacenters, nodes Moon and Leo, the service completion time was registered. The service begins when the first DCP message, *PE Info*, is received and it finishes when the CE router has been configured.

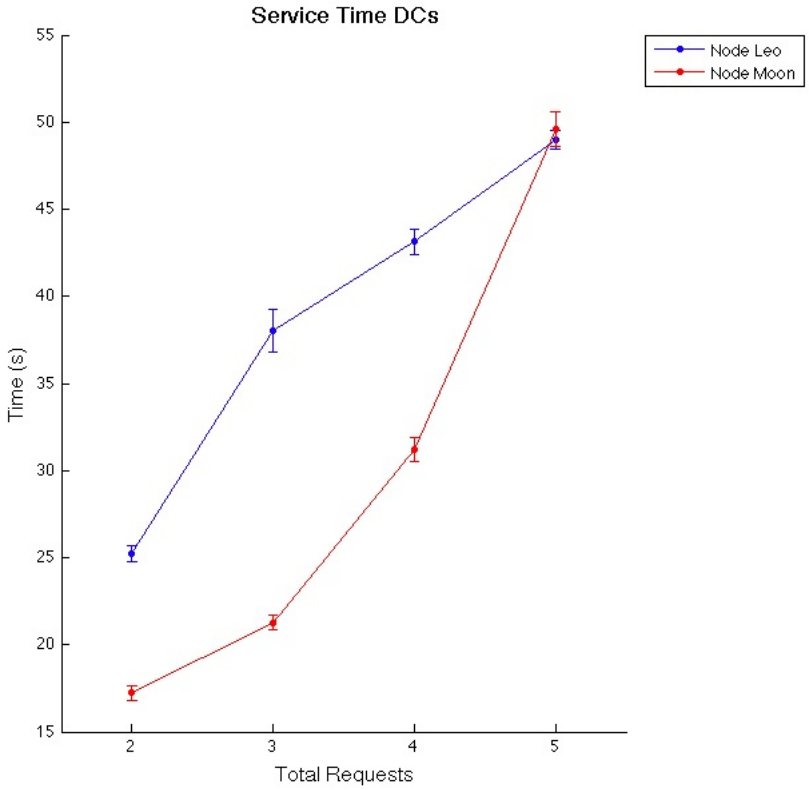


Figure 5.5: Total service time comparison between the two datacenters

In the request, the node Moon is defined before the node Leo, so it will be the first to be configured. This explains the difference registered between the two nodes (figure 5.5). This difference is only reduced in the last case, where 5 requests are sent simultaneously. These results were affected by the performance of the CE router connected to the node Moon. This router in some tests would stop responding and only in the second try a telnet session was established.

5.4 NVSS Tests

In this section the tests performed on the NVSS testbed are shown. The purpose of these tests is to assess the performance of the NVSS testbed regarding TCP and UDP traffic. These tests will allow the evaluation of different parameters: the TCP tests will give us an estimate on the impact of virtualization on the available bandwidth, while the UDP tests will reveal if there are any time related issues present. Depending on the results, this testbed may be integrated in the CloNe prototype in the future. This section starts with the TCP tests and afterwards it presents the UDP tests.

5.4.1 Testbed

The testbed used for the NVSS platform is composed of 6 physical nodes presenting the topology shown in figure 5.6. The specifications of the nodes can be seen in table 5.4.

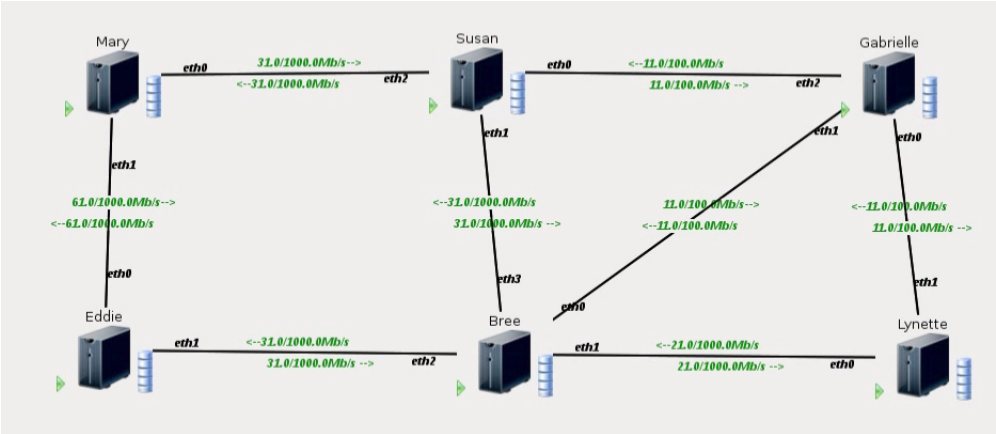


Figure 5.6: NVSS Testbed

Node	**Susan**	**Lynette**	**Gabrielle**	Bree	**Eddie**	**Mary**
CPU Model	Intel PentiumD 950	Intel PentiumD 950	Intel Core 2 Duo E6400	Intel Xeon E3110	Intel Xeon X3220	Intel Xeon X3330
CPU Freq	3.4GHz	3.4GHz	2.13GHz	3.0GHz	2.4GHz	2.66GHz
CPU Cores	2	2	2	2	4	4
CPU Threads	4	4	2	2	4	4
HDD Memory	89GB	40GB	145GB	195GB	303GB	277GB
RAM Amount	6GB	6GB	4GB	6GB	6GB	6GB
RAM Freq	533MHz DDR2	667MHz DDR2	533MHz DDR2	667MHz DDR2	667MHz DDR2	667MHz DDR2

Table 5.4: **NVSS - Machine Specification**

5.4.2 TCP Tests

The aim of this test is to measure the capabilities of the NVSS platform regarding throughput, ideally the performance should equal the performance of the physical interfaces. It is possible to observe the effect of varying the number of VRs and CPU load on the overall throughput.

Methodology

To perform these tests three nodes were used: Eddie as a transmitter; Susan as a receiver; and Bree where the VRs were mounted (see figure 5.7).

The three nodes are directly connected through ethernet cables with a bandwidth of 1000Mbps. To test the throughput the software *Iperf* was used which will allow to measure the transmission rate between the two nodes, Eddie and Susan. In all the tests 15 runs of 30 seconds of traffic were measured. The traffic is composed of TCP packets with a fixed

window size of 16KBytes.

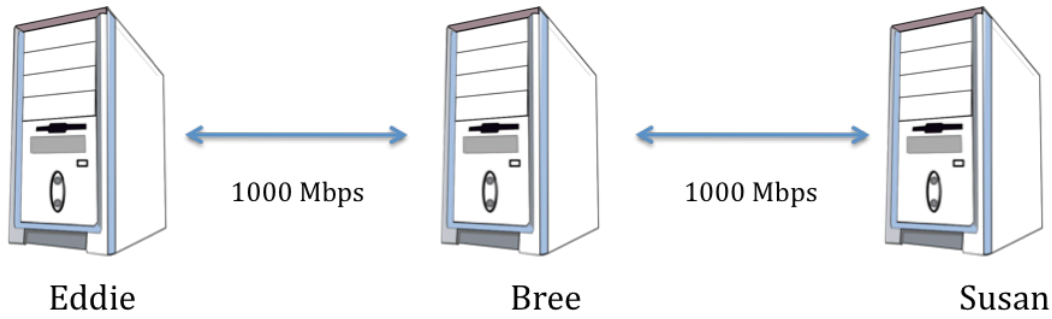
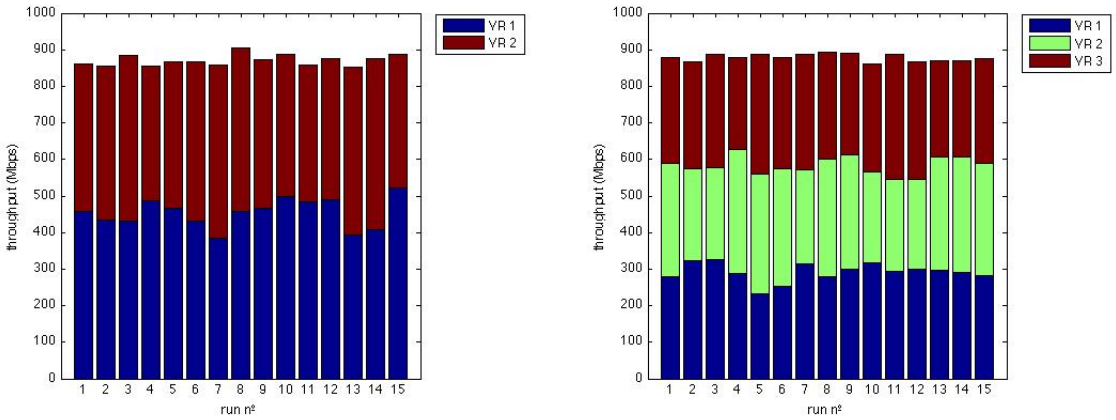


Figure 5.7: TCP Experimental Apparatus

Tests and Results

The first test measured the transmission rate between Eddie and Susan without the use of VRs, so a reference could be established to compare with the traffic measures with the VRs. The throughput registered was constant with a value of 940Mbps.

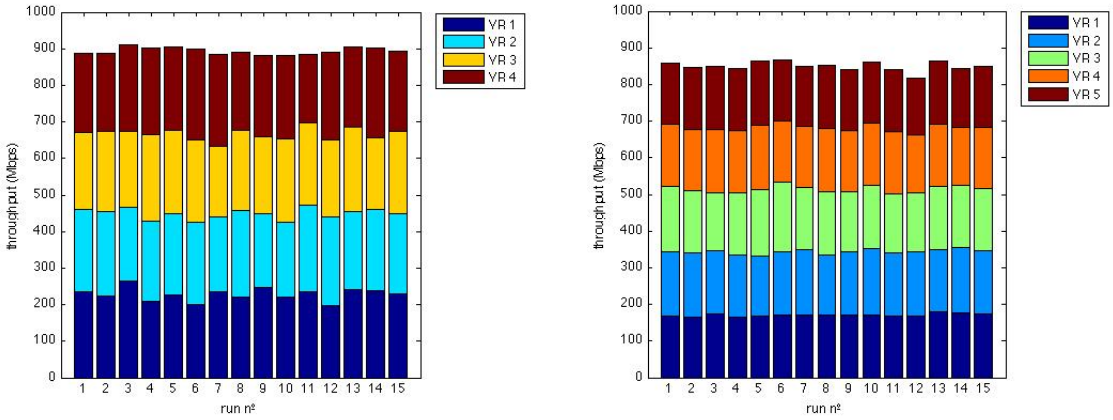
In the next series of tests the number of virtual routers used ranges from 2 to 7 with only one traffic flow per router (figures 5.8, 5.9 and 5.10). The aim of these tests is to evaluate the influence of the number of VRs on the transmission rate.



(a) Throughput with 2 VRs

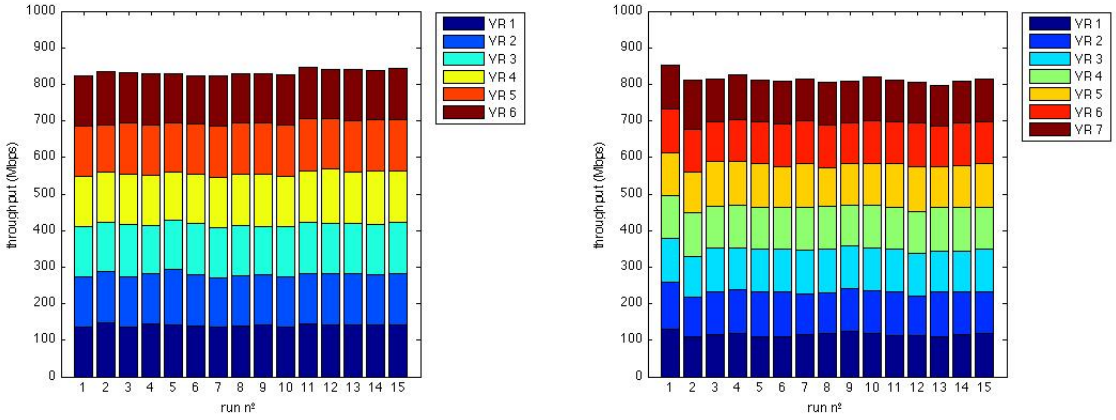
(b) Throughput with 3 VRs

Figure 5.8: TCP results for 2 and 3 VRs



(a) Throughput with 4 VRs (b) Throughput with 5 VRs

Figure 5.9: TCP results for 4 and 5 VRs



(a) Throughput with 6 VRs (b) Throughput with 7 VRs

Figure 5.10: TCP results for 6 and 7 VRs

In the table bellow we can see the standard deviation registered during the tests.

Number of VRs	**Standard deviation (Mbps)**
2	15.9484
3	14.9191
4	9.9162
5	9.87.69
6	12.6088
7	12.3912

Table 5.5: **TCP Standard Deviation**

The results show that the traffic is well distributed between the various routers, which

indicates that there is no problem with the increase of VRs in the same machine because the system makes a fair distribution of resources. Also the variation in the global throughput seems to be fairly stable during the test.

To assert the influence of the CPU load in the VRs performance, a program called *Lookbusy* was used. This software uses infinite loops in multiple threads to force the CPU load to a predetermined value. The results can be seen in figure 5.11.

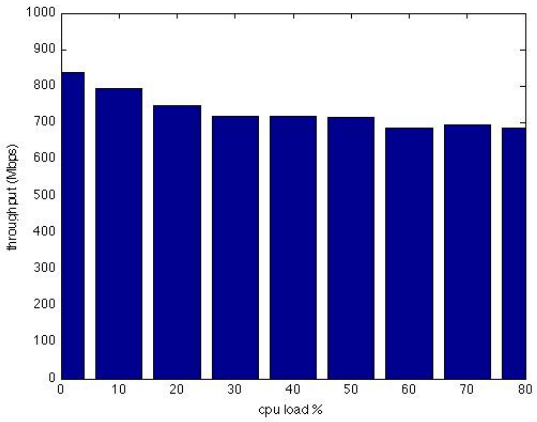


Figure 5.11: TCP CPU Load variation

The test showed that the CPU indeed affects the throughput value with a steady decrease until it reaches around 30% of CPU load, but after that the throughput seems to stabilize around 700Mbps.

5.4.3 UDP Tests

After some traffic tests evaluating the performance of virtual networks, some questions arose as to the jitter behavior in these networks. To assess this problem a reduced reference metric will be used, the Coefficient of Variation (CoV), instead of a full reference because of the limitations of the available testbed. By varying the number of VRs and flows per router, the CoV ratio between two distinct points shows any disturbance in the network in terms of uniformly distributed jitter. The testbed used is the same of the previous test.

Methodology

In this testbed three physical machines are used to generate traffic flows, and two physical machines to receive them. In the center there is one physical machine (Eddie, see table 5.4) with one VR per virtual network in use. To ensure independency between the various virtual links, VLAN tagging was used; the VLAN tagging is only applied between the switches. In the physical machine Eddie where the VRs reside each virtual interface is associated with a specific bridge and VLAN tag.

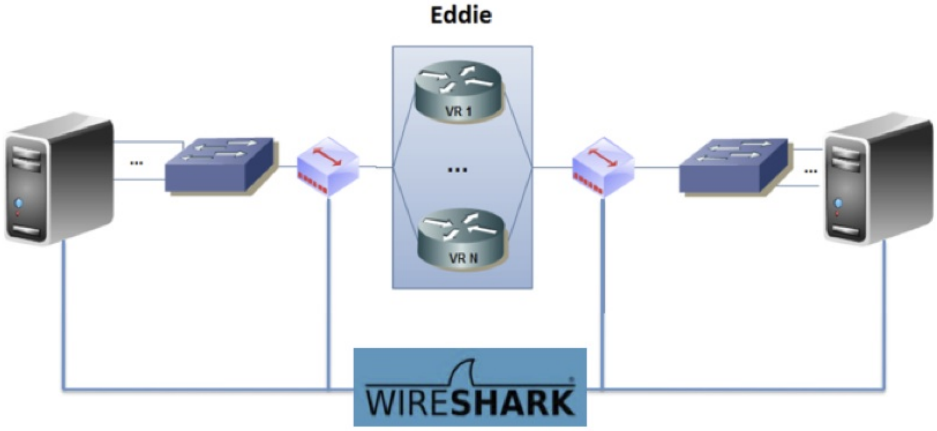


Figure 5.12: UDP Experimental Apparatus

Iperf [59] sessions were used to generate traffic flows with the following characteristics:

- size- 1300 bytes
- bit rate- 1Mbps

To have an *end2end* view of the packet transmission, the software Wireshark was used in four different points along the path (see figure 5.12):

1. In the machine generating the flow
2. In the hub before the node Eddie
3. In the hub after the node Eddie
4. In the machine receiving the flow

The wireshark application uses the machines system time to get the capture time of each packet.

To quantify the performance of the VRs we use the CoV in the four points of measurement. The CoV is defined by the equation:

$$CoV = \frac{standard\,deviation\,of\,throughput}{mean\,throughput}$$

The variable used to calculate the standard deviation and mean, is the number of packets in a timeframe that pass through each of the four points of measurement. The CoV value indicates the dispersion of this variable, the higher the value, the greater the dispersion. Using the CoV ratio between two points, we can obtain a clear indication of the impact of uniformly distributed jitter in the network. In the ideal case, when there is no jitter present, the CoV ratio is equal to one. If this ratio gets above ten, it means that there is jitter in the same order of magnitude of the observation interval [60]. In these tests it was used five timescales: 10 ms, 30ms, 100ms, 300ms, and 1s.

In the first part of the test, we have one VR with up to 6 flows of traffic, while in the second part we have up to six VRs with one flow of traffic per VR. The test results were similar in both cases, with the CoV ratio values remaining around 1 for up to 4 VRs and up to 4 flows in one VR. A CoV ratio value of 1 indicates that there is no time variations in the order of magnitude of the timescales in the packets delay. Therefore these results show that there is no jitter present in the order of magnitude of the observation interval (see figures 5.13, 5.14, 5.15, 5.16, 5.17 5.18).

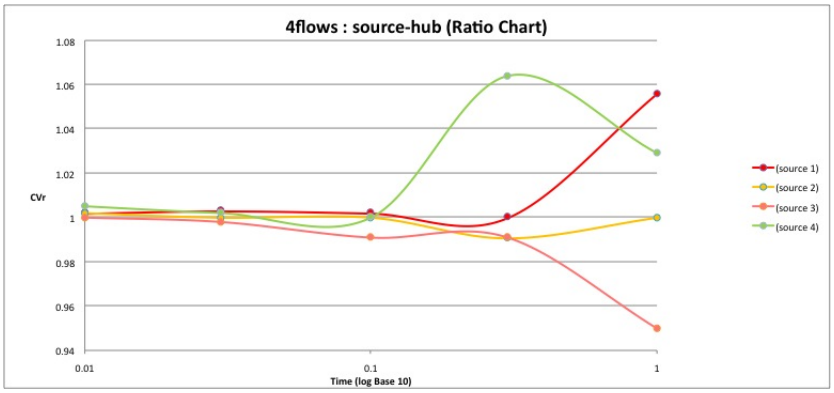


Figure 5.13: CoV between the source and the first hub with one VR and 4 flows

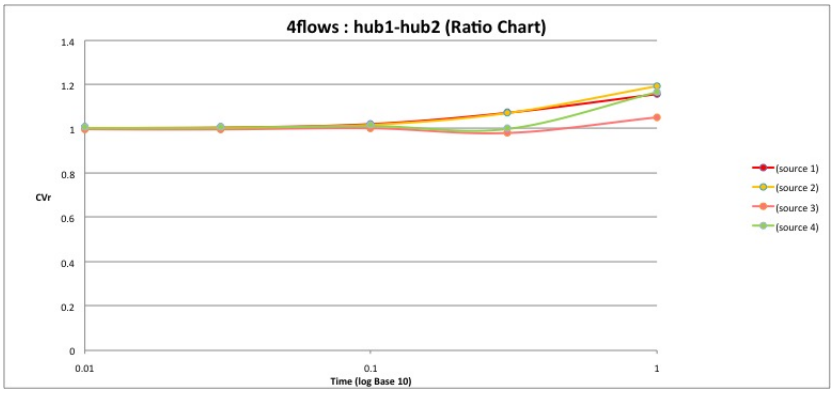


Figure 5.14: CoV between the two hubs with one VR and 4 flows

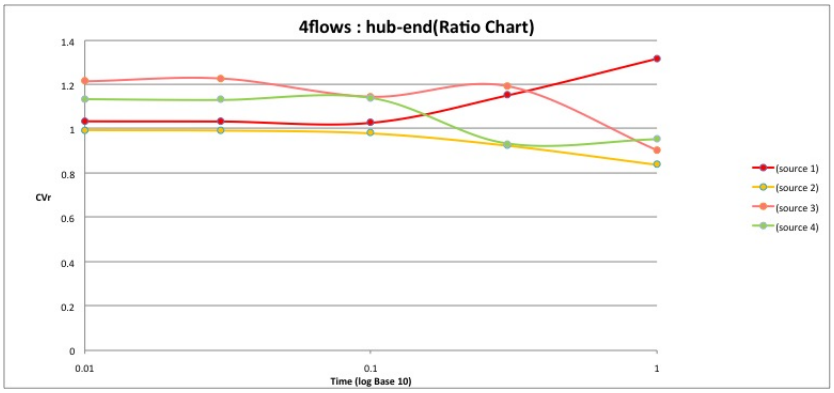


Figure 5.15: CoV between the second hub and the destination with one VR and 4 flows

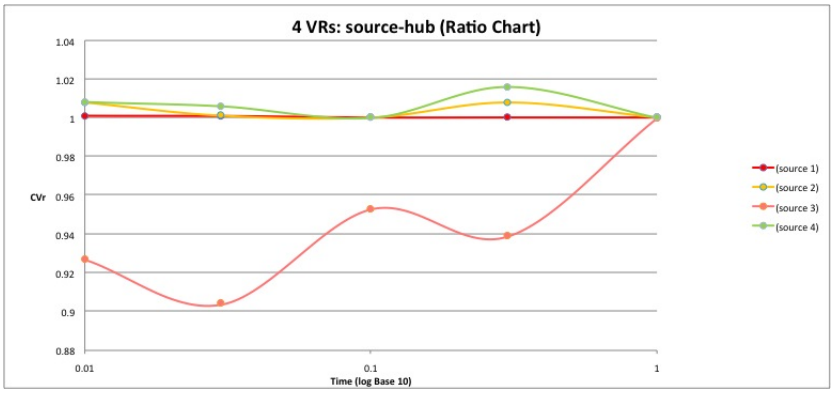


Figure 5.16: CoV between the source and the first hub with 4 VRs

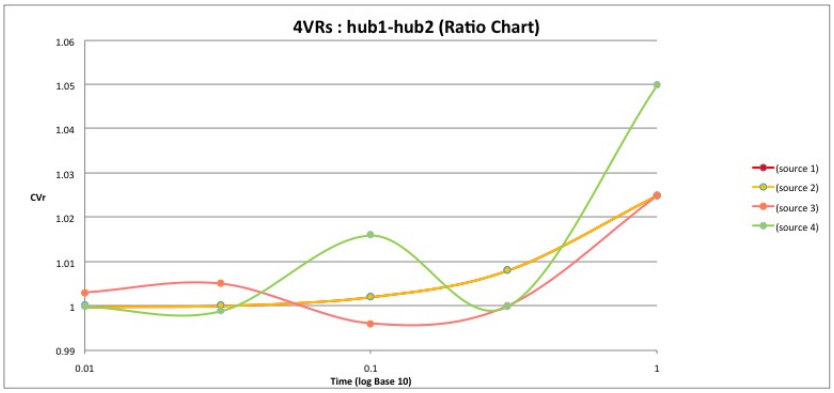


Figure 5.17: CoV between the two hubs with 4 VRs

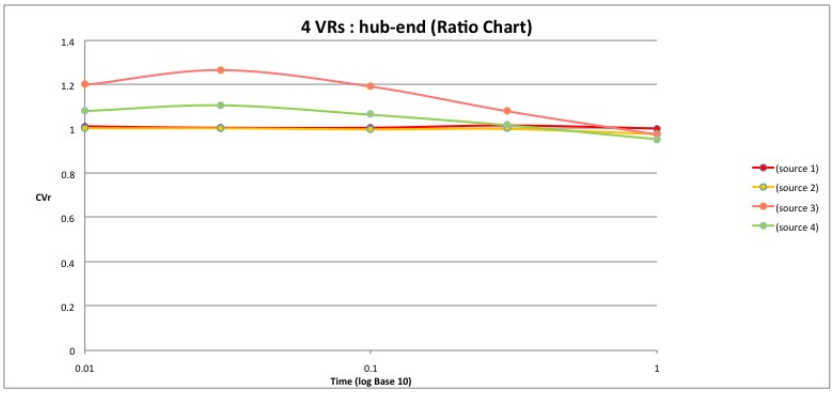


Figure 5.18: CoV between the second hub and the destination with 4 VRs

For more than four VRs and four flows the results get worse in the timescales between 0.1 and 1s, which indicates that the problem is not derived from individual packets delay variation, but due to interference operating within these timescales (see figures 5.19, 5.20, 5.21, 5.22, 5.23, 5.24). Although the traffic variation between the inlet and the outlet of the VRs is minimum, when comparing these two points with the source and destination respectively, it is possible to see an increase. This increase means that CoV values registered in the hubs are bigger than the ones registered at the source and destination; indicating that the packets transmission is suffering variations in the timescales comprehended between 0.1 and 1s.

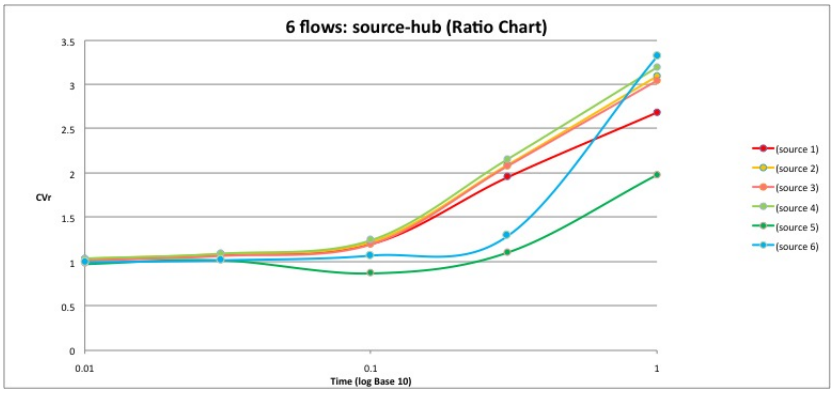


Figure 5.19: CoV between the source and the first hub with one VR and 6 flows

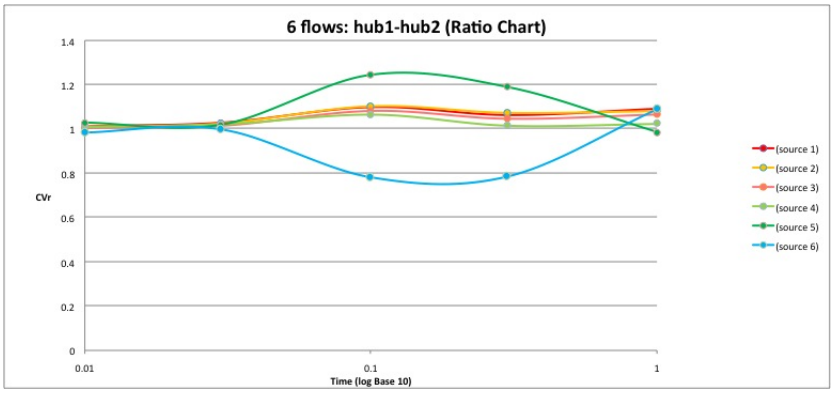


Figure 5.20: CoV between the two hubs with one VR and 6 flows

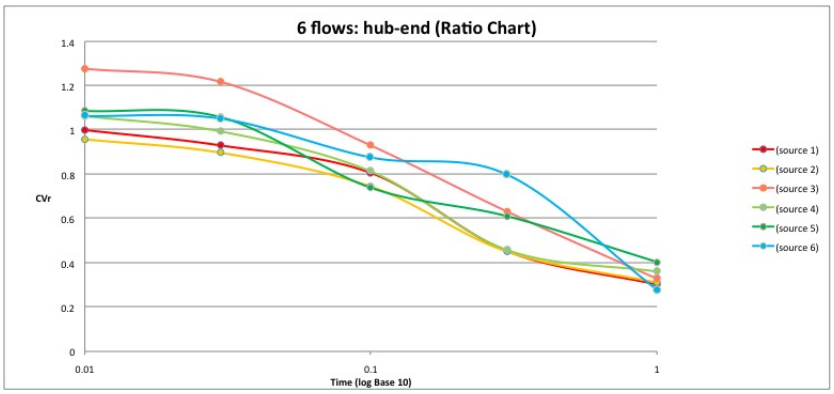


Figure 5.21: CoV between the second hub and the destination with one VR and 6 flows

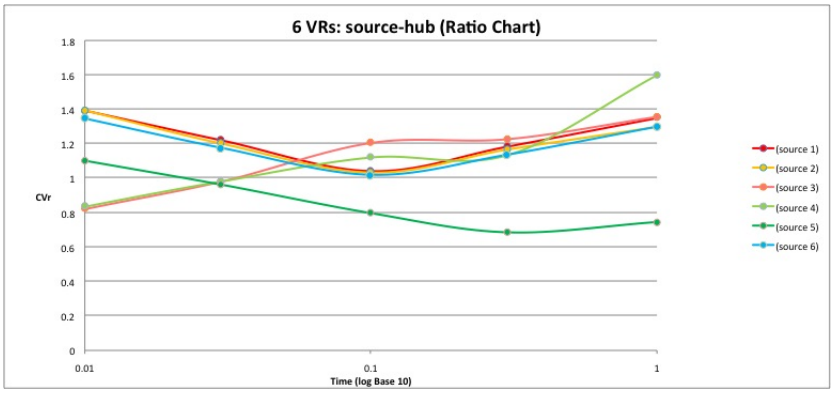


Figure 5.22: CoV between the source and the first hub with 6 VRs

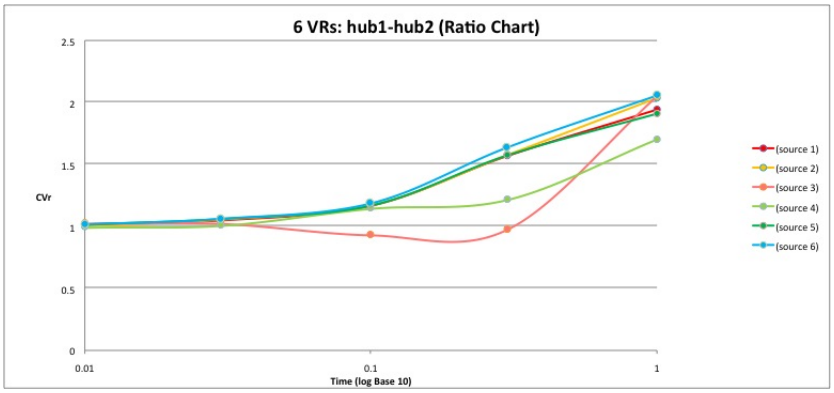


Figure 5.23: CoV between the two hubs with 6 VRs

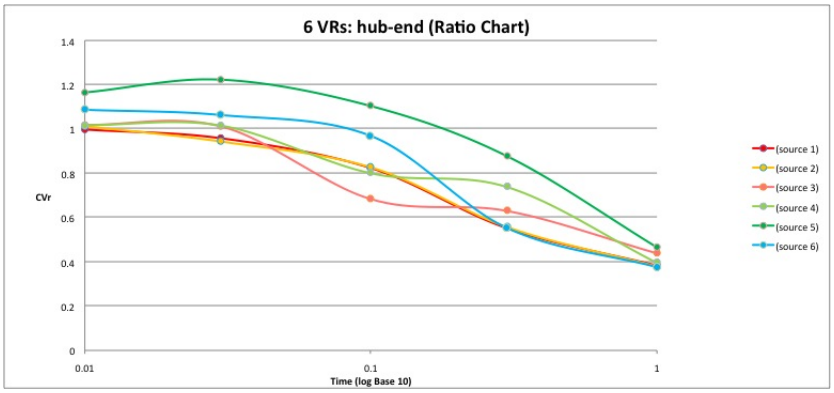


Figure 5.24: CoV between the second hub and the destination with 6 VRs

5.5 Conclusion

The analysis performed on the CloNe testbed gives an overview on the time necessary to deploy a CloNe service. In the NM analysis, the NA communication represents a possible bottleneck, which was expected due to the various steps involved in the operation:

- NM constructs the XML message and the properties file and saves them to disk
- NA client wraps the XML message in a SOAP message and sends it to NA
- NA parses the message and processes the request
- NA constructs and wraps a XML response message in a SOAP message and sends it back to the NA client

The comparison between the cloud domain and network domain in the CloNe analysis shows that the deployment of a CloNe service is largely dominated by the launching of the VM. It should be noted that today's datacenters have machines that are capable of deploying VMs in a fraction of this time. Even though the analysis on the Cloud Networking testbed is affected

by the underlying infrastructure, it clearly proves that CC services and WAN services can be instantiated within the same timescale in a single integrated request.

The results of the stress test performed on the NM showed that, even though the requests were sent simultaneously, it behaved like they were sent consecutively due to the presence of the lock mechanism. This mechanism increases the robustness of the NM with the trade off being the increased service completion time.

The tests performed in the NVSS platform showed a good performance for 4 VRs mounted in the same machine. In terms of throughput, only after 5 VRs it showed performance penalties with the increase of VRs, while the UDP tests showed that the packet delay remained constant for up to 4 VRs. Although the increase of CPU load immediately affected the throughput, its value stabilizes around 700Mbps when 30% of CPU load is reached; it should also be noted that having 7 VRs on the same machine never led the CPU load exceed the 10% threshold. The disturbances felt in the timescale between 0.1 to 1s in the UDP tests are not provoked by the presence of jitter, and are assumed to be resultant from the packet scheduling in the sender.

Chapter 6

Conclusion

6.1 Final Conclusion

The primary aim of this work is to deliver a prototype capable of delivering a composed cloud and WAN services. Although the testbed was limited in size, it was still possible to validate the concept, through the use of OCCI standard, the OCCI extension for network, OCNI, and the DCP.

Chapter 2 provided a further insight on CC, its characteristics, deployment and service models. Current offers of some of the most important Cloud providers were presented, as well as the current work developed by the most relevant standardization bodies in this area. The current platforms for CC and network slicing technologies were also presented in this area.

Chapter 3 starts by introducing the Cloud Networking architecture, in which the prototype is based on, as well as the proposed solution. The solution was decomposed to provide a better understanding of the mechanisms design and reference for the implementation chapter. The prototype dependencies, guidelines and features are also provided here.

In chapter 4 the implementation of the prototype was addressed, providing a better understanding of the approaches taken. This chapter covers the structure, sequence of actions and methods and classes used in the implementation of the three domains: the Cloud Networking provider domain; the Cloud Computing provider domain; and the Network Provider domain.

In Chapter 5 an analysis of the CloNe prototype is performed as well as a test in which simultaneous requests were made. The analysis performed on the Cloud Networking prototype proved that cloud computing resources and WAN resources can be instantiated within the same timescale. The results of the stress tests showed that the presence of the lock mechanism improves robustness at the expense of completion time. Traffic tests were also performed in the NVSS platform. These tests allowed a greater insight over its overall performance in which it was possible to achieve bandwidths near the physical limits using virtualized resources.

With this prototype, it is possible to deploy with a single request an infrastructure integrating computing and WAN resources, proving the importance of defining standards and common interfaces to improve interoperability between providers.

6.2 Future Work

The tests performed on the NVSS platform showed that further optimization should be done in terms of network performance. This platform can be integrated in the prototype either

on the Cloud domain, to extend the network capabilities of a datacenter, or in the Network domain to provide a better separation between FNSs and the underlying infrastructure. With that in mind an interface has to be created to allow the NM to interact with the manager module present in the NVSS in similar fashion as it is performed with the NA.

The CloNe prototype currently only supports basic functionalities, creation and removal of virtual infrastructures. It still lacks an update functionality, allowing to remove or change parts of the infrastructure. Before that function can be implemented, DCP messages have to be created to support it, and OpenNebula would have to be extended to support the update functionality.

The CloNe Orchestrator is still very limited. To extend its capabilities, a database is needed as well as a mean to exchange information with other providers regarding resource usage. By adding these two, advanced algorithms can be employed to instantiate CloNe services which is a fundamental requirement for the evolution of the prototype.

The tests performed on the NM module showed a side effect of the security mechanism used to protect its integrity. This mechanism hinders the ability to process multiple requests concurrently; an optimization of this mechanism should be done to reduce this side effect.

Appendix A

Tables

A.1 OpenNebula Tables

Table	Description
Group pool	identifies the group and the users who are part of it
User pool	identification of the user and also the password and the authentication driver
Image pool	identifies the storage resources and the associated permissions table. Also stores information about it's location, image type, image state and the associate VMs
Network pool	identifies the network resources and the associated permissions table. Also stores information regarding its type, the associated physical interfaces and IP addresses
Host pool	identifies the hosts and keeps an updated monitoring information
Vm pool	identifies the VM and the associated permissions table. Also stores information regarding the VM current state, the integrated virtual resources and history records

Table A.1: **ON Database Tables** - Open Nebula tables within the data base

A.2 OCCI Tables

Code	Description	Notes
200	OK	Indicates that the request was successfull. The response must contain the created resource instance's representation
201	OK	Indicates that the request was successfull. The response must contain the HTTP location header to the newly create resource instance
202	Accepted	Used for asynchronous non-blocking calls
204	OK, no content returned	This is used to indicate that a collecion is empty
400	Bad Request	Used to signal parsing errors or missing information.
401	Unauthorized	The client does not have the required permitions or credentials
403	Forbidden	Used to signal that a particular Mixin cannot be applied to a resource instance of a particular kind. Used to signal that an attempt was made to modify an attribute that was marked as immutable
404	Not Found	Used to signal that the request had information that was unknown to the service and so not found
405	Method Not Allowed	The service does not allow the client to issue the HTTP method against the requested path/location
406	Not Acceptable	When the client requests a Content-typpe that will result in an incomplete or faulty rendering
409	Conflict	A request contains content (e.g. mixin, kind, action) that results in an internal service, non-unique result. The client must resolve the conflict by re-trying with specific Category information in the request
410	Gone	A client attempts to retrieve a resource instance that no longer exists
500	Internal Server Error	The state before the request should be maintained in such an error condition. The imlementation must roll-back any partial changes made during the erroneous execution
501	Not Implemented	If an implemmentation chooses not to implement a particularOCCIfeature, it must signal the lack of that feature with this code. This implicity points to a non-compliantOCCIimplementation
503	Service Unavailable	If theOCCIservice is taken down for maintenance, this error code should be reported from the root of the name-space the provider uses

Table A.2: **HTTP Codes** - HTTP Return Codes [39]

Bibliography

[1] Ryan, Falvey, and Merchant, "Regulation of the cloud in india," *Journal of Internet Law*, vol. 15, no. 4, 2011.

[2] F. Dupre. (2012, June) Life in the Cloud, Living with Cloud Computing. [Online]. Available: http://computinginthecloud.wordpress.com/2008/09/25/utility-cloud-computingflashback-to-1961-prof-john-mccarthy/

[3] D. Craig. (2012, July) Construction Cloud Computing - Cloud Computing History 101. [Online]. Available: http://www.constructioncloudcomputing.com/2010/08/14/cloud-computing-history/

[4] Txchnologist - Online Magazine. (2012, July) Txchnologist - Douglas Parkhill, Author and Researcher. [Online]. Available: http://www.txchnologist.com/2011/7-thinkers-who-shaped-the-cloud/screen-shot-2011-10-28-at-4-58-17-pm

[5] A. Mohamed. (2012, June) Computer Weekly - A history of cloud computing. [Online]. Available: http://www.computerweekly.com/feature/A-history-of-cloud-computing

[6] Digital Marketing Solutions LLC. (2012, June) History of Salesforce. [Online]. Available: http://salesforceprogrammers.com/article467-History_of_Salesforce_.html

[7] (2012, June) Amazon Web Services. [Online]. Available: http://aws.amazon.com/pt/

[8] Cisco Systems, Inc. (2012, July) 2012 Cisco Global Cloud Networking Survey. [Online]. Available: http://www.cisco.com/en/US/solutions/ns1015/global_cloud_survey.html

[9] (2012, June) SAIL Project - Scalable and Adaptive Internet Solutions. [Online]. Available: http://www.sail-project.eu/

[10] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology, September 2011.

[11] B. Golden. (2012, June) CIO - Capex vs Opex: Most people miss the point about cloud economics. [Online]. Available: http://www.cio.com/article/484429/Capex_vs._Opex_Most_People_Miss_the_Point_About_Cloud_Economics

[12] (2012, June) Solutionext - A source for IT solutiones. [Online]. Available: http://www.solutionext.co.uk/

[13] D. Harris. (2012, June) Scoop: Google, Microsoft both targeting Amazon with new clouds - Cloud Computing News. [Online]. Available: http://gigaom.com/cloud/scoop-google-microsoft-both-targeting-amazon-with-new-clouds/

[14] (2012, June) AT&T Synaptic - AT&T Cloud Services. [Online]. Available: https://www.synaptic.att.com/

[15] (2012, June) SmartCloudPT - Serviços Cloud Computing da Portugal Telecom. [Online]. Available: http://www.smartcloudpt.pt/

[16] (2012, July) National Institute of Standards and Technology. [Online]. Available: http://www.nist.gov/index.html

[17] (2012, June) Distributed Management Task Force, Inc. [Online]. Available: http://dmtf.org/

[18] (2012, June) Open Grid Forum - Open Forum, Open Standards. [Online]. Available: http://www.gridforum.org/

[19] (2012, June) Storage Networking Industry Association - Advancing Storage and Information Technology. [Online]. Available: http://snia.org

[20] M. Carlson and M. McMinn, *Cloud Data Management Interface Extension: CIMI*, SNIA, January 2012.

[21] (2012, June) Cloud Security Alliance. [Online]. Available: http://cloudsecurityalliance.org

[22] (2012, June) The Internet Engineering Task Force. [Online]. Available: http://www.ietf.org/

[23] *Carrier Ethernet for Delivery of Private Cloud Services*, Metro Ethernet Forum, February 2012.

[24] (2012, June) Metro Ethernet Forum - Accelerating the Adoption of Carrier Ethernet. [Online]. Available: http://metroethernetforum.org/

[25] (2012, July) OpenStack - Open Source Cloud Computing Software. [Online]. Available: http://openstack.org

[26] (2012, June) OpenNebula - The Open Source Solution for Data Center Virtualization. [Online]. Available: http://opennebula.org

[27] Paul Murray et al., *Cloud Networking Architecture Description*, deliverable fp7-ict-2009-5-257448-sail/d.d.1 ed., SAIL project, July 2011.

[28] Network Working Group. (2012, July) Provider Provisioned Virtual Private Network Terminology. [Online]. Available: http://tools.ietf.org/html/rfc4026

[29] (2012, July) OpenFlow - Enabling Innovation in Your Network. [Online]. Available: http://www.openflow.org

[30] Open Networking Summit. (2012, May) Open Networkin Summit 2012 - why SDN? [Online]. Available: http://opennetsummit.org/why.html

[31] C. M. Salvador. (2012, June) Software-Defined Networking: Opportunities and Challenges.

[32] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento, "Network virtualization system suite: Experimental network virtualization platform," in *TridentCom 2011, 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2011.

[33] (2012, June) The FP7 4WARD Project. [Online]. Available: http://www.4ward-project.eu/

[34] (2012, June) GENI - Exploring Networks of the Future. [Online]. Available: http://www.geni.net/

[35] (2012, June) FP7 - EU's Seventh Framework Programme for Research. [Online]. Available: http://ec.europa.eu/research/fp7/

[36] (2012, June) euronf - Network of Excellence. [Online]. Available: http://euronf.enst.fr/en_accueil.html

[37] (2012, June) GEYSERS - Generalised Architecture for Dynamic Infrastructure Services. [Online]. Available: http://www.geysers.eu/

[38] (2012, June) Open Cloud Computing - Core. Open Grid Forum. [Online]. Available: http://www.gridforum.org/documents/GFD.183.pdf

[39] (2012, June) Open Cloud Computing - RESTful HTTP Rendering. Open Grid Forum. [Online]. Available: http://www.gridforum.org/documents/GFD.185.pdf

[40] (2012, June) Open Cloud Computing - Infrastructure. Open Grid Forum. [Online]. Available: http://www.gridforum.org/documents/GFD.184.pdf

[41] (2012, June) PyOCNI - A Python implementation of an extended OCCI with JSON serialization. [Online]. Available: https://github.com/jordan-developer/PyOCNI

[42] AMQP Working Group. (2012, June) Advanced Message Queing Protocol - Protocol Specification. [Online]. Available: http://www.amqp.org/sites/amqp.org/files/amqp.pdf

[43] T. Garnock-Jones and G. M. Roy. (2012, June) Pika - Pika 0.9.5 documentation. [Online]. Available: http://pika.github.com/

[44] (2012, June) Lyatiss - Cloud Network Control. [Online]. Available: http://www.lyatiss.com/

[45] (2012, June) Inria - Inventors for the Digital World. [Online]. Available: http://www.inria.fr/en

[46] M. Hellkamp. (2012, June) Bottle - Python Web Framework. [Online]. Available: http://bottlepy.org/docs/dev/

[47] (2012, May) SQLite - SQLite Home Page. [Online]. Available: http://sqlite.org/

[48] S. Ferg. (2012, June) Python Conquers The Universe - Python Decorators. [Online]. Available: http://pythonconquerstheuniverse.wordpress.com/2012/04/29/python-decorators/

[49] Secret Labs AB. (2012, June) ElementTree Overview. [Online]. Available: http://effbot.org/zone/element-index.htm

[50] (2012, July) cURL - Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/CURL

[51] springsource - A division of vmware. (2012, July) RabbitMQ - Messaging that just works. [Online]. Available: http://www.rabbitmq.com/

[52] Ruby Visual Identity Team. (2012, July) Linguagem de Programação Ruby. [Online]. Available: http://www.ruby-lang.org/pt/

[53] Edgewall Software. (2012, July) PySqlite - The Trac Project. [Online]. Available: http://trac.edgewall.org/wiki/PySqlite

[54] Python Software Foundation. (2012, June) SQLite Documentation. [Online]. Available: http://docs.python.org/library/sqlite3.html

[55] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. (2012, July) Simple Object Access Protocol (SOAP) 1.1. [Online]. Available: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[56] Cisco Systems, Inc. (2012, July) Cisco 7600 Series Ethernet over MPLS. [Online]. Available: http://www.cisco.com/en/US/products/hw/routers/ps368/prod_technical_reference09186a00800ae418.html

[57] ——. (2012, July) Cisco Network Virtualization. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns431/ns658/net_qanda0900aecd804a16ae.html

[58] P. Litwin. (2012, May) r937.com - Fundamentals of Relational Database Design. [Online]. Available: http://r937.com/relational.html

[59] (2012, July) Iperf - SourceForge project page. [Online]. Available: http://iperf.sourceforge.net/

[60] T. N. Minhas, M. Fiedler, and P. Arlos, "Quantification of Packet Delay Variation through the Coefficient of Throughput Variation," in *The 6th Internation Wireless Communications and Mobile Computing Conference IWCMC*, 2010.